مجدي محمد أبو العطا

المرجع الأساسي لقاعدة البيانات Clipper™

الجزء الأول

توزيع مؤسسة جمال الجاسم للالكترونيات
ص.ب ١٠٢ اللمام ٣١٤١١ ـ فاكس ٨٣٣٠٤٥١-٣-٩٦٦
تلفون ٨٣٣٢١٠٩ / ٨٣٢٢١٤٨

أجيز طباعة هذا الكتاب بموجب خطاب سعادة مدير عام المطبوعات بوزارة الإعلام بالمملكة العربية السعودية رقم ٣٩١٢/م بتاريخ ١٣/٦/١٤١١هـ.

الطبعة الأولى

١٤١١هـ - ١٩٩١م



المرجع الأساسي
لقاعدة البيانات
dBASE IV

يخاطب هذا الكتاب المبتدئين في إعداد نظم إدارة قواعد البيانات باستخدام قاعدة البيانات dBASE IV وذوي الخبرة الطويلة بالاصدارات السابقة من قاعدة البيانات "dBASE" مثل dBASE III PLUS.

فالجزء الأول من الكتاب يأخذ بيد القارىء خطوة خطوة من خلال تدريبات عملية مبسطة ليضع بين يديه أساسيات قاعدة البيانات dBASE IV والتي تتلخص في:

- مفهوم قواعد البيانات وتنظيم ملفاتها ومجالات استخدامها.
- إنشاء الملفات وإدخال بياناتها واستعراض محتوياتها والاستفسار عنها بشتى الطرق.
- ترتيب وتنظيم الملفات وإجراء العمليات الحسابية على بياناتها الرقمية.
- تصميم واستخراج التقارير والملصقات.

ويشرح الجزء الثاني البرمجة باستخدام قاعدة البيانات dBASE IV من خلال مجموعة كبيرة من البرامج معدة بطريقة تعليمية تتدرج من النظرية إلى التطبيق ومن الفهم إلى العمل مع التركيز على المفاهيم الجديدة والتي لم تكن موجودة بالاصدارات السابقة. ويشرح الجزء الثالث مفاهيم متقدمة في قاعدة البيانات تتضمن إعداد نظم شاملة باستخدام مصمم التطبيقات تستخدم التسهيلات التي أضافتها قاعدة البيانات «دي بيس فور». تعتبر نماذج حية يمكن الاقتداء بها لمن يريدون إعداد نظم مماثلة.

وإتماما للفائدة فقد اشتمل الكتاب على خمسة ملاحق هامة لا يستغني عنها أحد ممن يعملون في هذا المجال.



المرجع الأساسي

لقاعدة البيانات

dBASE III PLUS

يشرح هذا الكتاب كيفية استخدام قاعدة البيانات dBASE III PLUS مع الحاسبات الشخصية سواء من ناحية الأوامر واستخدام شاشات المساعدة أو من ناحية البرمجة.

والكتاب صيغ بأسلوب سهل ليخاطب أولئك المشتغلين في مجال الحاسبات ومن ليست لهم خبرة سابقة بالحاسبات الآلية فقد بدأ بشرح أساسيات واستخدامات الحاسبات الآلية في الجزء الأول قبل شرح قاعدة البيانات وأوامرها والتعامل معها. كما تم شرح أساسيات البرمجة في الجزء الثاني منه قبل شرح مفهوم واستخدام البرمجة في قاعدة البيانات.

ولذلك فقد جاء هذا الكتاب بحق مرجعاً أساسياً للمشتغلين والدارسين في هذا المجال. فقد كتب بأسلوب تعليمي منظم يصلح للتدريس في الجامعات والمعاهد العلمية. وقد جاء شاملاً أيضاً لكل ما تحتويه المادة ولكل ما يحتاج إليه العاملون في هذا المجال.



المرجع الأساسي

لمستخدمي لوتس ١-٢-٣

Lotus 1-2-3

يشرح هذا الكتاب واحدا من أقوى البرامج المتكاملة التي تتيح إعداد صفحة البيانات الالكترونية والرسوم البيانية وقواعد البيانات. والكتاب يخاطب المبتدئين وأصحاب الخبرة السابقة باستخدام برنامج LOTUS 1-2-3® (لوتس ١-٢-٣). فبدأ بتقديم نظرة عامة عن برامج صفحة البيانات الالكترونية وبرنامج LOTUS 1-2-3® بصفة خاصة. ثم تدرج في شرح جميع الامكانيات الأخرى التي تيسر إعداد صفحة البيانات الالكترونية وإعداد وطباعة الرسوم البيانية وبناء قواعد البيانات والمختزلات. أما أصحاب الخبرة السابقة باستخدام البرنامج فسيجدون فائدة عظيمة من خلال التمارين العملية التي يشتمل عليها الكتاب والتي تزيدهم فهما لامكانيات البرنامج، وتعتبر نماذجا حية يمكن الاسترشاد بها في حياتهم العملية. فهذه التمارين تشرح جميع الأوامر اللازمة لاعداد التطبيقات المتكاملة التي تشتمل على صفحة البيانات الالكترونية والرسوم البيانية وقواعد البيانات والمختزلات (MACROS).

وإتماما للفائدة تناول الكتاب كيفية استخدام البرنامج مع البيانات العربية باستخدام جميع برامج تعريب مدخلات ومخرجات الحاسب الموجودة بالساحة حتى الآن.

تقديـــم

الحمد لله والصلاة على سيدنا رسول الله وعلى آله وصحبه . وبعد،

هذا هو الكتاب السابع في **سلسلة تيسير علوم الحاسب** التي تميزت بالدقة العلمية وسهولة العرض وشمولية المادة ووفرة التمارين العملية. ويأتي هذا الكتاب ليسد لبنة هامة وضرورية لمبرمجي قواعد البيانات لأن الكتاب لا يقف عند شرح ترجمة وربط برامج «دي بيس» فقط بل يتناول أيضا الموضوعات التالية:

■ البرمجة باستخدام «كلبر».

■ المصفوفات

■ شبكات الاتصالات

■ التعامل مع أخطاء البرامج

■ تطبيقات شاملة تعتبر نماذج حية يستفاد بها في إعداد نظم إدارة قواعد البيانات.

■ مرجع شامل لجميع الأوامر والوظائف.

لذلك فإننا نعتبر أن إصدار هذا المرجع يعتبر إنجازا علميا نتوقع أن يستفيد منه الكثيرون من مبرمجي «دي بيس» و«كلب» والله من وراء القصد.

مصطفى الصيني

مدير مركز أبحاث شبكة الكمبيوتر الشخصي

الكتاب في سطور

تم تقسيم كتاب المرجع الأساسي لقاعدة البيانات Clipper إلى أربعة أبواب وثلاثة ملاحق على النحو التالي:

الباب الأول... مفاهيم أساسية

الفصل الأول. . . مدخل إلى قاعدة البيانات Clipper. وهو يبدأ بمقدمة تاريخية عن قواعد البيانات عموما ثم مقدمة تاريخية عن Clipper وضرورتها. والفرق بين المفسر الموجود في dBASE والمترجم الموجود في Clipper.

الفصل الثاني. . . الخصائص المميزة لبرنامج Clipper. ويتناول أهم خصائص ومكونات Clipper مثل: متطلباتها. . . حدودها. . . أنواع الملفات والحقول التي تستخدمها. . . العلامات الحسابية والمنطقية التي تستخدمها. . وأخيرا التعبيرات.

الفصل الثالث. . . تركيب وتهيئة Clipper. ويشرح تركيب «كلبر» ـ الحصول على معلومات مساعدة ـ تجهيز ملفات AUTOEXEC.BAT و CONFIG.SYS.

الفصل الرابع. . . ترجمة برامج dBASE III PLUS باستخدام Clipper. ويشتمل على الأوامر والوظائف التي لا يتعامل معها Clipper والموجودة في dBASE III PLUS والأوامر والوظائف الجديدة في Clipper والتي لم تكن موجودة في dBASE III PLUS ليسهل عليك تعديل برامجك القديمة أو إعداد برامجك الجديدة في ضوء الفروق الموجودة بينهما.

الفصل الخامس. . . البرمجة باستخدام Clipper. ويشرح الملامح الجديدة في «كلبر». والتسهيلات التي يجب أن يعرفها مستخدمو «كلبر» لأول مرة مثل: ملف الاجراءات ـ الوظائف الخاصة ـ المصفوفات ـ إعداد القوائم ذات الشريط المضاء ـ استخدام التعبيرات بدلا من اختيارات بعض الأوامر ـ استخدام مفاتيح الوظائف لاستدعاء

برنامج أو إجراء ـ التعامل مع ملفات خارجية ـ استخدام أمر FOR...NEXT لانشاء دوارة.

الفصل السادس... ترجمة البرامج وربطها مع نظام التشغيل. ويعتبر هو العمود الفقري للكتاب لأنه يشرح طريقة ترجمة (Compiling) البرامج المصدرية إلى برامج هدف (Object) وربطها (Linking) مع بعضها لاستخراج ملف جاهز للتنفيذ (.EXE).

الباب الثاني... مفاهيم متقدمة

الفصل السابع... المصفوفات: إنشاؤها ـ تعبئتها ـ نسخها ـ حذفها ـ ترتيب عناصرها ـ البحث فيها.

الفصل الثامن... مكتبة «كلبر» ويوضح أهم ملفات مكتبة «كلبر» بالاضافة إلى شرح كيفية استخدام برامج المساعدة التي تأتي مع حزمة «كلبر» مثل برامج إنشاء وتصميم ملفات قواعد البيانات وملفات التقارير والملصقات وملفات الفهرسة وبرنامج استخدام «كلبر» من نقطة توجيه الأوامر (Dot-Prompt).

الفصل التاسع... تعقب وتصحيح أخطاء البرامج ويوضح كيفية ربط مكتشف الأخطاء مع البرنامج واستدعائه وكيفية استخدامه لتعقب أخطاء البرنامج واكتشافها.

الفصل العاشر... التعامل مع أخطاء البرامج ويوضح كيفية التعامل مع الأخطاء الناتجة عن الملفات أو التعبيرات أو الطباعة... الخ.

الفصل الحادي عشر... استخدام «كلبر» مع شبكة الاتصالات. ويبدأ بتوضيح الفرق بين إعداد نظم إدارة قواعد بيانات لتستخدم بواسطة مستفيد واحد (Single user) أو بواسطة مجموعة مستفيدين (Multiusers) ثم يشرح كيفية تلافي المشاكل التي قد تنجم عن استخدام أكثر من مستفيد لنفس البيانات في نفس الوقت بالاضافة إلى شرح الأوامر والوظائف التي تستخدم بصفة خاصة في برامج شبكات الاتصالات.

الباب الثالث... تطبيقات شاملة باستخدام «كلبر»

الفصل الثاني عشر. . ويشتمل على نظام كامل لادارة قاعدة البيانات يتكون من مجموعه كبيرة من البرامج والاجراءات والوظائف الخاصة يمكن أن تستخدم بحالتها الراهنة أو بعد تعديلها ـ حسب حاجتك ـ لاعداد نظم إدارة قواعد بيانات مماثلة. والنظام يصلح لخدمة مستفيد واحد أو أكثر من مستفيد مرتبطين داخل شبكة اتصالات. وإتماما للفائدة شرحنا كيفية تعريب النظام.

الباب الرابع... مرجع الأوامر والوظائف

الفصل الثالث عشر. .مرجع الأوامر ويشتمل على شرح شامل لجميع أوامر «كلبر» ويتناول كل أمر من ناحية: وظيفته ـ الشكل العام ـ شرح الاختيارات التي يشتمل عليها ـ شرح الأمر والوصايا المقترحة لاستخدامه ـ الفرق بينه وبين dBASE III PLUS ـ مثال على الأقل ـ المكتبة الموجود بها ـ الأوامر الأخرى ذات الصلة.

الفصل الرابع عشر. . .مرجع الوظائف ويشتمل على شرح شامل لجميع الوظائف الموجودة بمكتبات «كلبر» ويتناول شرح كل وظيفة الموضوعات الآتية: شرح مختصر ـ الشكل العام ـ شرح اختياراتها ـ شرح عام للوظيفة ـ اختلافها عن dBASE III PLUS ـ مثال على الأقل ـ المكتبة التي توجد بها ـ الأوامر والوظائف ذات الصلة.

الملاحق

الملحق الأول. . . ملخص المصطلحات والرموز

الفصل الثاني. . . الشفرة الأمريكية لتبادل المعلومات

الفصل الثالث. . . أهم المكتبات وبرامج الخدمات التي يمكن الاستفادة منها في إعداد نظم إدارة قواعد البيانات.

المحتويات

الفصل الخامس

الفصل السادس

الفصل السابع

الفصل الثامن

الفصل التاسع

الفصل العاشر

الفصل الحادي عشر

الباب الثالث: تطبيقات شاملة

مقدمـة

إن الحمد لله، نحمده ونستعينه ونستهديه ونستغفره ونشكره، ونصلي ونسلم على خير خلقه وخاتم رسله محمد بن عبدالله وعلى آله وصحبه أجمعين.

﴿سبحانك لا علم لنا إلا ما علمتنا، إنك أنت العليم الحكيم. . .﴾ وبعد

لقد ترك النجاح الذي حققته مجموعة كتب قواعد البيانات التي كتبناها ضمن **سلسلة تيسير علوم الحاسب** والذي إن دل على شيء فإنما يدل على وعي القارىء العربي وحرصه على متابعة الجديد في تكنولوجيا الحاسبات ترك في نفسي إحساسا بالمسئولية تولد عنـه التفكير في كتابة مجموعة جديدة من الكتب تخاطب الذين استفادوا من سلسلة الكتب التعليمية التي بدأناها مع بداية انتشار الحاسبات الشخصية. وكان هذا الكتاب هو الأول من هذه النوعية الجديدة والتاسع في **سلسلة تيسير علوم الحاسب**. وهو يشرح أشهر وأفضل مترجم للغة dBASE «دي بيس» ليس فقط بل أيضا يمكن اعتباره أقوى قاعدة بيانات يمكن استخدامها في تطوير نظم إدارة قواعد البيانات.

وقد تم تصميم «كلبر» كلغة برمجة متوافقة مع «دي بيس ثري بلس» مع إمكانية ترجمة (Compiling) البرامج لتعمل ذاتيا تحت نظام التشغيل.

إن «كلبر» تتميز على «دي بيس ثري بلاس» بمجموعة من الأوامر والوظائف والتسهيلات تجعلها بحق أقوى وأفضل الحزم البرامجية التي طورت من أجل إعداد نظم إدارة قواعد البيانات باستخدام لغة dBASE.

ورغبة في دقة تحديد المصطلحات سنوضح الفرق بين أصطلاح dBASE «دي بيس» وعبارة dBASE III PLUS «دي بيس ثري بلاس» (الآن dBASE IV).

اصطلاح dBASE كلمة عامة تطلق على لغة البرمجة الشهيرة التي تم تطويرها في بداية الثمانينات بواسطة شركة «اشتن تيت» الأمريكية. والتي بدأت بـ dBASE II ثم تلاها كثير من البرامج مثل: FoxBase, DbXI, Quickselver ويعتبر Clipper أحدها. أما dBASE III PLUS أو dBASE IV فتخص قاعـدة البيانات التي أنتجتها شركة

«أشتن تيت». وقد يطلق اصطلاح dBASE ويقصد به منتجات شركة «أشتن تيت» التي بدأت بـ dBASE II وانتهت بـ dBASE IV.

لمن هذا الكتاب

الكتاب يصلح لمن لديهم فكرة عن قاعدة البيانات dBASE III PLUS أو غيرها من البرامج المتوافقة معها وسبق لهم كتابة برامج باستخدامها. سواء كانوا مبتدئين أم ذوي باع طويل في البرمجة. فللمبتدئين يشرح لهم الحاجة إلى ترجمة برامج dBASE والفرق بين المترجم (Compiler) والمفسر (Interpretor) الموجود في «دي بيس ثري بلاس» وكيفية ترجمة البرامج التي في حوزتهم وربطها مع نظام التشغيل والفرق بين «دي بيس ثري بلاس» و«كلبر». ولأصحاب الخبرة السابقة بالبرمجة يشرح المصفوفات وكيفية استخدام «كلبر» مع شبكات الاتصالات وكيفية التعامل مع أخطاء البرامج وكذلك تعقب واكتشاف الأخطاء التي قد يقعون فيها. ولكل من الفريقين زودنا الكتاب بمرجع كامل لجميع الأوامر والوظائف بالاضافة إلى تطبيقات شاملة تعتبر نماذج حية يستفاد منها في حياتهم العملية.

والكتاب ييسر لذوي الخبرة بقاعدة البيانات dBASE III PLUS (أو غيرها من قواعد البيانات المتوافقة مثل FoxBase أو DBXL) الانتقال إلى قاعدة البيانات Clipper في غضون بضعة أيام بالرغم من الفهم السائد عند البعض أن استخدام Clipper أصعب من استخدام dBASE III PLUS ولعل السبب في هذه المقولة أن كلا من البرنامجين من إنتاج شركة غير الأخرى وأن «دي بيس ثري بلاس» أسبق من «كلبر».

محتويات الكتاب

يشتمل هذا الكتاب على أربعة أبواب وثلاثة ملاحق. يبدأ الباب الأول بإعطاء مقدمة عن قواعد البيانات عموما وقاعدة البيانات Clipper بصفة خاصة. ثم يشرح الحاجة إلى «كلبر» وكيفية البرمجة باستخدام Clipper. ثم يوضح لمستخدمي dBASE III PLUS كيفية تطوير برامجهم باستخدام Clipper. وأخيرا كيفية ترجمة (Compiling) البرامج وربطها (Linking) مع نظام التشغيل. أما الباب الثاني فيعطي نظرة موسعة لمستخدمي «كلبر» تشمل البرامج التي تشتمل عليها مكتبة «كلبر». وكيفية استخدام

المصفوفات وتعقب وتصحيح أخطاء البرامج وكيفية التعامل مع أخطاء البرامج. ولمستخدمي «كلبر» مع شبكة الاتصالات يشرح استخدام «كلبر» مع شبكة الاتصالات. والباب الثالث يورد تطبيقات شاملة باستخدام «كلبر» والباب الرابع يشتمل على مرجع شامل لجميع الأوامر والوظائف التي تشتمل عليها «كلبر» تصلح لخدمة مستفيد واحد أو مجموعة مستفيدين داخل شبكة اتصالات.

ولذلك فإن موضوع هذا الكتاب مستوى لا يستغني عنه المبرمجون الذين يعدون تطبيقاتهم لتنافس أعلى مستوى في عالم اليوم.

مصطلحات الكتاب

يشتمل هذا الكتاب على العديد من المصطلحات التي لا يوجد لها مرادف في قواميس اللغة. وقد اجتهدت رأيي في وضع مرادف لهذه المصطلحات يعطي المعنى المقصود بدقة. وإليك بعض الأمثلة:

- استخدمنا كلمة «كلبر» لأنها علم على اسم البرنامج موضوع الكتاب Clipper.
- استخدمنا كلمة ترجمة مقابل كلمة compiling وكذلك كلمة مترجم مقابل كلمة Compiler.
- استخدمنا كلمة الربط مع نظام التشغيل مقابل كلمة linking.
- استخدمنا كلمة قاعدة البيانات بدون إضافة ونقصد بها قاعدة البيانات Clipper.
- استخدنا كلمة حقل الذاكرة مقابل عبارة Memory Variable.

وبعد عزيزي القارىء فهذا جهد قليل من كثير قصد به النفع للمسلمين وأبناء العرب جميعا. وخطوة على الطريق تتلوها خطوات بإذن الله. فإن وجدت منه فائدة فلا تنسانا من دعوة صالحة. وإن كانت غير ذلك فيسعدني تلقي اقتراحاتك وتوجيهاتك ووضعها في الحسبان في الطبعات والكتب القادمة بإذن الله.

وآخر دعونا أن الحمد لله رب العالمين.

مجدي محمد أبوالعطا

الباب الأول

مفاهيم أساسية

الفصل الأول
مدخل إلى قاعدة البيانات Clipper

يعتبر هذا الفصل مدخلا إلى قاعدة البيانات Clipper ويبدأ بمقدمة تاريخية عن قواعد البيانات عموما. والفرق بين كل من dBASE و dBASE III PLUS ثم مقدمة تاريخية عن تاريخ «كلبر» ثم يشرح ضرورة «كلبر» والفرق بين المفسر الموجود في dBASE والمترجم الموجود في Clipper.

مقدمة تاريخية

كان تشغيل قواعد البيانات في الماضي ينفذ فقط على الحاسبات الكبيرة وذلك لحاجتها إلى مكان كبير للتخزين ومعالج قوي للبيانات (Powerful Processor). ولكن التقدم السريع في تكنولوجيا الحاسبات خفض من تكلفة الحاسب. بالاضافة إلى اختراع حاسبات ذات ذاكرات كبيرة. وبذلك أمكن تشغيل قواعد البيانات على الحاسب الشخصي. وقد بدأ تشغيل قواعد البيانات على الحاسبات الشخصية في بداية الثمانينات بقاعدة البيانات dBASE II التي اشترتها شركة «أشتن تيت» الأمريكية وكانت توزع تحت اسم فولكان (Vulcan). وطورتها بعد ذلك إلى dBASE III ثم dBASE III PLUS وأخيرا DBASE IV.وكان إنتاج برنامج dBASE III PLUS هو البداية التي دعت كثيرا من الشركات إلى المنافسة في إنتاج قواعد بيانات شبيهة ومتوافقة معها. وحاولت كل شركة أن تضيف إلى إنتاجها شيئا يجعله متميزا ومطلوبا.

ونورد فيما يلي على سبيل المثال لا الحصر قواعد البيانات المتوافقة والمتشابهة مع قاعدة البيانات dBASE III PLUS

- قاعدة البيانات FoxBASE وهي من إنتاج شركة «فوكس سوفتوير» الأمريكية وطورت إلى +FoxBASE وأخيرا FoxPro وهي أكثر قواعد البيانات توافقية مع dBASE III PLUS وتتميز عنها بالسرعة وزيادة بعض الإمكانيات مثل القوائم والمصفوفات والوظائف الخاصة.
- قاعدة البيانات dBXL وهي من إنتاج شركة «وورد تيتش سيستمز»، وهي أيضا من أكثر قواعد البيانات توافقية مع dBASE III PLUS وتتميز عنها باستخدام النوافذ والرسوم البيانية والمصفوفات والوظائف الخاصة.
- قاعدة البيانات Quickselver وهي من إنتاج شركة «كويك سلفر سوفتوير» واشترت حقوقها شركة «وورد تيتش سيستمز» وهي عبارة عن مترجم للغة «دي بيس» متوافق إلى حد كبير مع «دي بيس ثري بلاس» ومتوافق تماما مع «دي بي إكس إل» وبإمكانه استخراج برامج جاهزة للتنفيذ (EXE).

إلا أنه لا يشتمل على مفسر (Interpreter) مثل «دي بيس ثري بلاس».

- قاعدة البيانات Clipper وهي من إنتاج شركة «نانتوكيت» الأمريكية وأخذت شهرة عظيمة نظرا للامكانيات الهائلة التي تشتمل عليها والتي تعتبر إضافة إلى قاعدة البيانات dBASE III PLUS مثل استخدام المصفوفات والوظائف الخاصة والقوائم ذات الشريط المضاء (light bar) وتحسينات كثيرة على حقل الملاحظات (Memo). بالاضافة إلى قدرتها على استخراج ملف جاهز للتنفيذ (.EXE)

الفرق بين dBASE و dBASE III PLUS

ورغبة في دقة تحديد المصطلحات سنوضح الفرق بين الاصطلاح dBASE «دي بيس» وعبارة dBASE III PLUS «دي بيس ثري بلاس» (dBASE IV الآن).

الاصطلاح dBASE كلمة عامة تطلق على لغة البرمجة الشهيرة التي تم تطويرها في بداية الثمانينات بواسطة شركة أشتن تيت الأمريكية والتي بدأت بـ dBASE II ثم تلاها كثير من البرامج المتوافقة معها مثل:

FoxBASE, dBXL, Quickselver, Clipper

أما DBASE III PLUS أو dBASE IV فتخص قاعدة البيانات التي أنتجتها شركة «أشتن تيت». وقد يطلق الاصطلاح dBASE ويقصد به منتجات شركة «أشتن تيت» فقط والتي بدأت ببرنامج dBASE II وانتهت ببرنامج dBASE IV

تاريخ «كليبر»

يقولون إن الحاجة أم الاختراع. ولذلك فقد بدأ التفكير في إنتاج برنامج يترجم البرامج المكتوبة بلغة «دي بيس» إلى صورة قابلة للتنفيذ (.EXE) وقد بدأت الفكرة من قبل مجموعة من المبرمجين الذين اشتركوا في إنتاج قاعدة البيانات الشهيرة dBASE III PLUS وتبنى الفكرة مدير المجموعة «براين راسل» وعرضها على «بيري ريبل» وهو صاحب إحدى الشركات التي ساهمت بمبرمجيها في تطوير dBASE III PLUS. وعقد بينهما اجتماع على شاطىء «ماليبو» في استراحة اسمها «نانتوكيت» وفي هذا الاجتماع

قررا إنشاء شركة لانتاج البرنامج واختاروا له اسم «كلبر». وأخذ الاسم من إحدى اللوحات الموجودة على أحد جدران الاستراحة. وسميت الشركة فيما بعد «نانتوكيت» على اسم الاستراحة التي اجتمعا فيها. وقد نال البرنامج شهرة منقطعة النظير لأنه أضاف كثيرا من الامكانيات والتسهيلات إلى برنامج dBASE III PLUS بالاضافة إلى ميزة الترجمة التي انفرد بها ويتضح ذلك من الفقرة التالية.

ضرورة «كلبر»

نوجز فيما يلي أهم المزايا التي ينفرد بها Clipper والتي تجعله ليس فقط مترجما للغة «دي بيس» (dBASE Compiler) بل أيضا وسيلة قوية لتطوير نظم إدارة قواعد البيانات:

- الهدف الأساسي من تطوير برنامج Clipper هو تمكين تشغيل البرامج التي طورت باستخدام dBASE III PLUS (أو Clipper) بصفة مستقلة تحت محث نظام التشغيل (DOS) بدون حاجة لتشغيل برنامج dBASE III PLUS أو برنامج Clipper معها. وبهذا تتوفر المساحة التي يتطلبها أي من البرنامجين من ذاكرة الحاسب عندما نحتاج لتنفيذ تطبيقات معدة بأي منهما. وبالتالي تقتصر الحاجة على نسخة واحدة وهي النسخة التي يتم بها تطوير البرنامج أو النظام لأول مرة فقط.
- حماية البرامج المصدرية (Source Code) من الاطلاع عليها أو تعديلها من غير ذوي الاختصاص أو الجهات غير المسئولة. فبالرغم من سهولة قراءة البرامج في صورتها المصدرية. والصورة المصدرية هي الصورة التي يكتب بها البرنامج بشيفرة ASCII فيصعب أو يستحيل قراءة البرنامج بعد ترجمته إلى الصورة القابلة للتنفيذ (Executable Form).
- تشتمل «كلبر» على مجموعة من الأوامر والوظائف غير موجودة في «دي بيس ثري بلاس» والتي أضيفت فيما بعد إلى «دي بيس فور» لا يستغني عنها الذين يحتاجون لتطوير برامج وأنظمة قوية مثل استخدام الوظائف الخاصة (User Defined Functions) أو المصفوفات (Arrays). أو القوائم ذات

الشريط المضاء أو وظائف حقل الملاحظات (Memo).

- يمكن كتابة برامج صغيرة أو أجزاء منها (Modules) باستخدام لغات البرمجة القوية «سي» C أو «أسمبلي» Assembly وربطها مع البرامج المكتوبة بواسطة «كلبر» لتصبح في النهاية برنامجا واحدا.
- تقتصر أوامر «دي بيس» التي لا يمكن استخدامها ضمن برامج «كلبر» على الأوامر التي تستخدم شاشة كاملة وتسمى Full Screen Commands. وهذه الأوامر يندر استخدامها ضمن البرامج. ولذلك فلن تبذل جهدا كبيرا في ترجمة البرامج والتطبيقات التي طورتها باستخدام dBASE III PLUS.
- أضافت الشركة المنتجة لبرنامج Clipper معظم التسهيلات الموجودة في dBASE III PLUS مثل إنشاء وتعديل ملفات قواعد البيانات وملفات الفهرسة وإعداد التقارير المطلوبة عن طريق برامج موجودة في حزمة «كلبر» تؤدي هذه التسهيلات بطريقة مشابهة لمثيلتها في «دي بيس».

المترجم والمفسر

من الأسباب التي جعلت لقاعدة البيانات dBASE III PLUS قبولا لدى الناس سهولة تعلمها وإعداد تطبيقات كاملة باستخدامها بالاضافة إلى أنها لاتحتاج إلى تعلم لغة برمجة خارجية. إلا أنها تعتمد في تنفيذ البرامج على مفهوم وجود المفسر (Interpreter) الذي يقرأ الأوامر أمرا أمرا. فإذا وجد الأمر صحيحا انتقل إلى الأمر التالي. أما إذا اكتشف خطأً فإنه يتوقف عن التنفيذ ويظهر رسالة لتدل على نوع الخطأ.

وبظهور dBASE IV حلت هذه المشكلة بترجمة البرامج المكتوبة بقاعدة البيانات وتسمى (source programs) إلى لغة الآلة (object code). وتتلخص فكرة الترجمة (compiling) في مراجعة كل ال... التي يشتمل عليها البرنامج مرة واحدة. فإذا كانت كلها صحيحة يتم ترجمة ... المصدر (source) من لغة «دي بيس» إلى لغة الآلة. ولغة الآلة عبارة عن رموز ...ها الحاسب فقط. لكن بقيت مشكلة لدى أولئك الذين يودون بيع برامجهم أو ت... ا بعيدا عن قاعدة البيانات التي استخدمت

في تطوير البرامج وهي حاجتهم إلى برامج في صورة جاهزة للتنفيذ وهو ما يطلق عليه .EXE

وسنوضح فيما يلي بشيء من التفصيل الفرق بين المترجم (Compiler) والمفسر (Interpreter).

المفسر Interpreter

عبارة عن ملف يشتمل على جميع أوامر اللغة ولعل أقرب مثال لذلك مفسر لغة بيسك. فلكي تنفذ برنامجا مكتوبا بلغة بيسك يجب أن تستدعي أولا مفسر البيسك (بعض الشركات الآن تضع مفسر لغة بيسك ضمن الذاكرة RAM أثناء تصنيع الحاسبات) ثم تحمل البرنامج المكتوب بلغة بيسك في الذاكرة ثم تطلب من بيسك تنفيذ هذا البرنامج (بإصدار أمر RUN مثلا).

ويتم مراجعة كل سطر موجود بالبرنامج على حدة ليتأكد المفسر أن الأمر موجود ضمن الأوامر المحفوظة عنده وأن تركيبه صحيح وأنه موافق لقواعد اللغة.

فإذا كان الأمر موافقا لهذه الشروط يتم قبوله وتحويله إلى لغة الآلة لكي يفهمه الحاسب. أما إذا اكتشف المفسر خطأ بالبرنامج في أحد السطور فإنه يتوقف عن تنفيذ هذا السطر ويظهر رسالة عن نوع الخطأ على الشاشة. وعليك أن تصحح الخطأ الذي حصلت عليه وتعيد تنفيذ البرنامج مرة ثانية. ويستمر هذا الاجراء حتى يصير البرنامج خاليا من الأخطاء.

والعيب في هذه الطريقة أن البرنامج الذي يتم اختباره وتنفيذه يعاد اختباره بواسطة المفسر سطرا سطرا في كل مرة نطلب فيها إعادة تنفيذ هذا البرنامج. ولهذا يتضاعف وقت تنفيذ البرنامج مرات عديدة.

المترجم Compiler

باستخدام المترجم يتم مراجعة أوامر البرنامج أمرا أمرا للتأكد من خلوها من الأخطاء ـ كما يفعل المفسر ـ والفرق بينهما أن المترجم يقرأ جميع أوامر البرنامج ويظهر

قائمة بالأخطاء الواردة في جميع الأوامر مرة واحدة. وعندما يتم تصحيح جميع أخطاء البرنامج فإن المترجم يترجم الأوامر المكتوبة إلى لغة الآلة. ويضع الناتج على ملف جديد يشتمل على الأوامر بعد ترجمتها إلى لغة الآلة ويسمى برنامج الهدف (object code). وعندما تحتاج لتنفيذ البرنامج في المرة القادمة فإن التنفيذ يتم من برنامج الهدف (object code). وبذلك توفر وقت ترجمة البرنامج أمرا أمرا إلى لغة الآلة مادام البرنامج لم يتغير[1]

الربط مع نظام التشغيل Linking

بعد ترجمة برنامج المصدر (source code) إلى برنامج الهدف (object code) كما يحدث باستخدام مترجم dBASE IV أو FoxBase لا يمكن تشغيله بصفة مستقلة تحت نظام التشغيل(DOS) والسبب في ذلك أن البرنامج المترجم (object) يحتاج إلى إجراء آخر وهو ربطه مع نظام التشغيل وتسمى هذه العملية linking ويسمى البرنامج الذي يقوم بها linker أو Linkage editor ولذلك فإن البرامج التي يتم ترجمتها باستخدام قواعد البيانات مثل dBASE IV أو FoxBase تستخدم برنامجا مساعدا يسمى Run-time يتولى ربط البرنامج الذي يخصص له الاسم الممتد DBO. (معناها DataBase Object). والناتج بواسطة مترجم dBASE IV أو Fox. (معناها Foxbase) والناتج بواسطة مترجم FoxBase. يتولى ربط البرنامج مع نظام التشغيل.

وتعتبر إمكانية الربط مع نظام التشغيل (linking) العامود الفقري لبرنامج Clipper والميزة التي تميزه عن غيره من البرامج التي تستخدم لتطوير قواعد البيانات. والبرنامج الذي يتولى عملية الربط مع نظام التشغيل يسمى Plink86 plus [2]

---

(١) سوف نعرف في الفصل السادس كيف تم ترجمة البرامج باستخدام Clipper

(٢) ستعرف فيل الفصل السادس كيف يتم ربط البرنامج/البرامج مع نظام التشغيل باستخدام Plink86 plus

بعد ربط البرنامج مع نظام التشغيل (linking) بواسطة Clipper يتم إنشاء ملف جاهز للتنفيذ يخصص له الاسم الممتد (EXE.) وهذا الملف يشتمل على كل ما نحتاجه لتنفيذ البرنامج من تحت نظام التشغيل بدون حاجة لتشغيل قاعدة البيانات سواء كانت dBASE III PLUS أو Clipper وكل ما تحتاج إليه هو أن تكتب اسم البرنامج تحت محث نظام التشغيل وسيتولى الحاسب تنفيذه بسرعة هائلة تصل إلى عشرة أضعاف سرعة تنفيذ برنامج المصدر لأننا لسنا في حاجة لمراجعة قواعد اللغة (Interpreting) أو ربط البرنامج مع نظام التشغيل (linking)

ونود التنبيه إلى أن برنامج الهدف (object) يجوز ربطه مع نظام التشغيل منفردا أو مع برامج أخرى في صورة object سواء كانت مكتوبة باستخدام Clipper أو C أو Assembly.

ولا يمكن تعديل البرنامج بعد ترجمته إلى لغة الآلة فإذا أردت تعديل البرنامج يجب أن ترجع إلى النسخة الأصلية وتعدل فيها ثم تعيد ترجمة البرنامج وربطه مع نظام التشغيل مرة ثانية.

تـذكـر...!

تناولنا في هذا الفصل أهم البرامج الموجودة في السوق والتي تستخدم لتطوير نظم إدارة قواعد البيانات مع الحاسبات الشخصية والفروق بينها وأوضحنا أن «كلبر» تمتاز عنها جميعا بمجموعة من الأوامر والوظائف التي تضفي قوة على التطبيقات المعدة بها بالاضافة إلى ميزة ترجمة البرامج وربطها معا ومع نظام التشغيل لاستخراج ملف جاهز للتنفيذ (EXE.) تحت DOS وشرحنا الفرق بين المفسر (Interpretor) وهو البرنامج الـذي يتولى مراجعة الأوامر أثناء تنفيذها ليتأكد من صحتها أمرا أمرا كلما استُدعي البرنامج للتنفيذ والمترجم (Compiler) الذي يقوم بمراجعة كل أوامر البرنامج مرة واحدة ويظهر قائمة بكل الأخطاء الواردة بالبرنامج.

الفصل الثاني

الخصائص المميزة

لبرنامج Clipper

يشرح هذا الفصل أهم خصائص ومكونات قاعدة البيانات Clipper التي يجب أنْ تعرفها قبل أن تبدأ التعامل معها مثل:

- متطلباتها
- حدودها
- أنواع الملفات
- أنواع الحقول
- العلامات الحسابية والمنطقية التي تستخدمها
- التعبيرات

متطلبات Clipper (Requirements)

لكي تستخدم Clipper يجب أن يتوفر لديك الأجهزة والبرامج التالية:

١ - حاسب شخصي أي . بي . إم من نوع PC أو XT أو AT أو PS/2 أو أي جهاز متوافق معه .

٢ - ذاكرة لا تقل عن ٢٥٦ ك . ب .

٣ - قرص مرن وقرص آخر صلب.

٤ - نظام تشغيل (DOS) إصدار 2.0 أو أكثر للاستخدام الفردي أو نظام تشغيل (DOS) إصدار 3.1 أو أكثر في حالة استخدام شبكة اتصالات محلية .

٥ - ويمكن حسب اختيارك إضافة الملحقات التي تراها ضرورية لك مثل الطابعة أو المعالج الحسابي Math Co.processor.

حدود Clipper (Limitations)

١ - يمكن أن يشتمل الملف الواحد حتى ١ بليون سجل .

٢ - طول السجل وعدد الحقول في السجل مفتوح في حدود حجم الذاكرة (RAM).

٣ - حتى ١٥ ملفا مفهرسا يمكن فتحها في وقت واحد . وتستخدم Clipper نوعين من ملفات الفهرسة :

أ - NDX. وهو النوع الذي تستخدمه dBASE III PLUS

ب - NTX. وهذا النوع من الفهرسة خاص بكلبر.

٤ - الحد الأقصى لعدد حقول الذاكرة (Memory Variables) هو ٢٠٤٨ حقلا .

٥ - الحد الأقصى لعدد المصفوفات هو ٢٠٤٨ مصفوفة . لأن كل مصفوفة تعد بحقل ذاكرة واحدة ويمكن أن تشتمل المصفوفة الواحدة حتى ٤٠٩٦ عنصرا .

٦ - حتى ٢٥٥ ملفا يمكن فتحها معا .

٧ ـ يمكن تقسيم الذاكرة إلى ٢٥٤ منطقة (Workareas) ويمكن أن نضع حتى ١٥ ملفا مفهرسا في المنطقة الواحدة.

٨ ـ ملفات الاجراءات (procedure files) والوظائف الخاصة (User Defined Functions) ستعرفها فيما بعد ـ يمكن ترجمتها ضمن البرامج. وبذلك تصبح جزءا من البرنامج نفسه. وفي هذه الحالة لا داعي لفتح ملف الاجراءات.

٩ ـ حتى ٨ ملفات فرعية يمكن ربطها مع الملف الأصلي.

أنواع ملفات Clipper (File Types)

تستخدم «كلبر» نفس الملفات التي تستخدمها «دي بيس ثلاي بلاس» باستثناء ملفات الفهرسة فيمكنك اختيار النوع الذي تستخدمه «دي بيس» أو الذي تستخدمه «كلبر» ويتم إنشاء هذه الملفات إما بالطريق المعروف باستخدام dBASE III PLUS أو باستخدام برنامج DBU الذي يأتي ضمن برامج Clipper. وإليك شرح هذه الملفات باختصار.

١ ـ DBF (Data Base File).

وهو الملف الرئيسي الذي يشتمل على البيانات التي تنوي الاحتفاظ بها ويشتمل هذاا لملف على البيانات في شكل حقول (fields) داخل سجل (record) حيث يشتمل كل سجل على بيانات مختلفة وعادة يكون طول السجل ثابتا.

ويتم إنشاء ملف قاعدة البيانات إما بالطريقة المألوفة باستخدام قاعدة البيانات dBASE III PLUS، أو باستخدام برنامج DBU، الذي يأتي ضمن حزمة برامج Clipper. ولانشاء هذا البرنامج يجب تشغيل ملف تجميعي اسمه MAKEDBU.BAT يأتي أيضا ضمن حزمة برامج «كلبر» وذلك بكتابة أمر MAKEDBU تحت الدليل الذي توجد عليه ملفات «كلبر» ثم ضغط مفتاح الادخال. (راجع الفصل الثامن من هذا الكتاب).

ملاحظة: ننصح بعدم تشغيل هذا البرنامج قبل قراءة الفصل السادس: ترجمة البرامج وربطها، لأنك قد تحتاج لتعديل محتويات الملفات حسب أسماء الأدلة التي تشتمل على برنامج «كلبر» ومكتباته.

ونود الاشارة هنا إلى أن إنشاء ملف قاعدة البيانات (.DBF) باستخدام برنامج DBU يسمح بزيادة عدد الحقول في السجل الواحد وعدد الحروف في الحقل الواحد كما سيتصح ذلك بعد قليل.

٢ ـ .DBT (DataBase memo file)

يشمل البيانات التي أدخلت إلى حقل الملاحظات (memo field) الموجود في الملف ويمكن أن يحتوي الملف حتى ٦٤ ك.ب. لكل سجل.

٣ ـ .FMT (Format file)

يحتوي هذا الملف على أوامر SAY...@ التي تستخدم لاظهار شاشة إدخال أو تعديل البيانات. وتتعامل Clipper مع هذا الملف شأنه شأن ملفات البرامج الأخرى بدون حاجة لفتحه أو إغلاقه كما هو الحال في dBASE III PLUS.

٤ ـ .FRM (Report file)

تشتمل حزمة برامج Clipper على برنامج اسمه RL.EXE (بمعنى Report Label) يستخدم لتصميم ملف التقارير أو الملصقات.

ويجب تنفيذ برنامج MAKERL.BAT للحصول على ملف RL.EXE الذي يسمح بتصميم ملف التقرير أو الملصقات ويتم حفظ التقرير الذي ينشأ بهذه الطريقة على ملف FRM. (راجع الفصل الثامن من هذا الكتاب).

٥ ـ .LBL (Label file)

هذا الملف يشتمل على تصميم شكل الملصقات ويتم إنشاء ملف الملصقات (.LBL) بواسطة ملف RL.EXE الذي يستخدم لانشاء التقارير.

٦ ـ NDX (Index files). أو NTX.

يمكن حسب اختيارك استخدام ملف NDX. المعروف في dBASE III PLUS للفهرسة. أو ملف الفهرسة الموجود في Clipper باسم NTX. ويتميز ملف NTX. الذي يتعامل معه Clipper بأنه أسرع من NDX. ولذلك فإذا كنت تنوي استخدام برامجك بواسطة Clipper فقط فننصحك بالتعامل مع ملف NTX.

وكلا الملفين يستخدم لترتيب سجلات الملف الأصلي حسب حاجة المستخدم.

٧ ـ MEM (Memory file)

وهو ملف يشتمل على بعض القيم التي يتم تخزينها مؤقتا بذاكرة الحاسب أثناء تنفيذ البرنامج. ويمكن تخزين هذه القيم من الذاكرة إلى القرص الممغنط كما يمكن استرجاعها من القرص الممغنط إلى الذاكرة. ولا يوجد فرق بين هذا الملف وملف MEM. الذي تستخدمه dBASE III PLUS.

٨ ـ PRG (Program and Procedure files).

يشتمل على أوامر Clipper التي تُكوِّن في مجموعها برنامجا مكتوبا بشفرة ASCII ويمكن كتابة البرنامج بأي منسق للنصوص يتعامل مع شفرة ASCII وملفات الاجراءات (procedures) تعامل معاملة ملف البرنامج أي أنها لاتحتاج لفتح أو إغلاق كما هو الشأن في dBASE III PLUS.

٩ ـ TXT (Alternate file).

ملف يشتمل على بيانات مكتوبة بشفرة ASCII ويمكن استخدام هذه البيانات بواسطة برامج أخرى.

أنواع الحقول (Field Types)

١ ـ حرفي Characteric

يقبل أي حرف قابل للطباعة ويمكن إدخاله من لوحة المفاتيح ويشمل الحروف والأرقام والمسافات الخالية والعلامات الخاصة وأقصى طول له هو ٣٢ ك.ب. ولا

يمكن إجراء عمليات حسابية على محتوياته حتى ولو كانت أرقاما كما في حالة تسجيل رقم الهاتف أو رقم الموظف في حقل حرفي.

٢ ـ رقمي Numeric

يشتمل على الأرقام التي ستجرى عليها عمليات حسابية ومن الممكن أن يكون كله رقم صحيح أو عشري وأقصى طول له ١٩ خانة مشتملة الاشارة والعلامة العشرية إن وجدت ومن أمثلة الحقل الرقمي الحقل الذي يشتمل على راتب الموظف أو بدل انتقاله.

٣ ـ تاريخي Date

دائما طوله ٨ أحرف ويأخذ الشكل mm/dd/yy ومعناه رقمين من اليسار يسجل فيهما الشهر يليهما رقمان يسجل فيهما اليوم يليهما رقمان تسجل فيهما السنة وهو الاستخدام الأمريكي للتاريخ. ويمكن إجراء العمليات الحسابية على الحقول التاريخية.

٤ ـ منطقي Logical

دائما طوله حرف واحد ويقبل فقط T أو F (True/False) أو البديل لها Y أو N (Yes/No) يمعنى نعم أو لا.

٥ ـ ملاحظات Memo

يستخدم لتسجيل كمية كبيرة من النصوص تصل إلى ٦٤ ك. ب. بطريقة بعيدة عن قيود قاعدة البيانات مثل معلومات عن سيرة الموظف الذاتية وبيانات حقل الملاحظات تسجل في ملف مستقل من نوع dbt. وتميز في ملف قاعدة البيانات بكلمة Memo. ويتطلب ١٠ خانات في سجل قاعدة البيانات.

العلامات الحسابية والمنطقية التي تستخدمها Clipper
(Clipper Operators)

أولا : علامات حسابية *(Mathematical Operators)*

تستخدم «كلبر» العلامات الحسابية المتعارف عليها في معظم البرامج لاجراء العمليات الحسابية المختلفة وهي :

+ للجمع   - للطرح   * للضرب   / للقسمة   ** لرفع الأس (القوة)

ويتم تنفيذ العمليات الحسابية التي تشتمل على هذه العلامات طبقا للأولويات التالية :

١ - يتم تنفيذ المعادلة أو المسألة الحسابية من اليسار إلى اليمين.

٢ - يتم التخلص من الأقواس الداخلية (إن وجدت) ثم الأقواس الخارجية.

٣ - يتم تنفيذ عمليات رفع القوة.

٤ - يتم تنفيذ عمليات الضرب والقسمة (* أو /).

٥ - يتم تنفيذ عمليات الجمع أو الطرح (+ أو -).

ثانيا : علامات تربط المتغيرات بعلاقة معينة *(Relational Operators)*

وتستخدم لعقد مقارنات بين قيمتين لمعرفة هل هما متساويتين أم مختلفتين أم أن إحداهما أكبر أو أصغر من الأخرى وهذه العلامات هي :

< أصغر من   > أكبر من   = يساوي   <> أو # أو != لا يساوي

<= أصغر من أو يساوي   >= أكبر من أو يساوي

$ علامة وجود أو تطابق عبارتين   == يساوي.

ثالثا : علامات منطقية *Logical Operators*

تستخدم لمعرفة حالة ما. هل هي صحيحة أم خاطئة وتستخدم داخل البرنامج للتفريع من تعليمة إلى أخرى وهذه العلامات هي :

.AND.	للبحث عن بيان يشترك فيه كلتا صفتين. أي يجب أن يكون كل من الشرط المذكور قبلها والشرط المذكور بعدها صحيحا ليكون التعبير صحيحا.
.OR.	للبحث عن بيان يشتمل على إحدى صفتين. أي إذا وقع أحد الشرطين المذكورين قبلها أو بعدها صحيحا يعتبر التعبير صحيحا.
.NOT. أو !	للبحث عن بيان لا يشتمـل على صفة معينة. أي إذا لم يقع الشرط المذكور بعدها صحيحا فيكون التعبير صحيحا.

ويتم تنفيذ هذه العلامات طبقا للأولويات التالية:

١ ـ التخلص من الأقواس الداخلية (إن وجدت) ثم الأقواس الخارجية.
٢ ـ تنفيذ علامة .NOT.
٣ ـ تنفيذ علامة .AND.
٤ ـ تنفيذ علامة .OR.

رابعا: علامات ربط العبارات (String Operators)

١ ـ علامـة + تربط جملتـين (تعبيرين) أو أكثر مع بعضهما بما في ذلك الفراغات لتكون جملة واحدة في النهاية.
٢ ـ علامـة – تربط جملتين أو أكثر مع بعضهما وتحذف الفراغات الموجودة بعدها لتكون جملة واحدة في النهاية.

التعبيرات (Expressions)

تستخدم التعبيرات (expressions) بنفس المفهوم الذي تستخدم به في dBASE III PLUS. ويمكن أن يشتمل التعبير الواحد على بيانات حقل (Data field) أو حقل ذاكرة (Memvar) أو قيمة ثابتة (Constant) أو ناتج معادلة أو بعض أو كل هذه المكونات.

ويمكن أن يكون التعبير واحدا من الأنواع الأربعة التالية :

١ ـ حرفيا (expC) مثل "NCC" أو TRIM(FIRSTNAME)

٢ ـ رقميا (expN) مثل 2+5*4

٣ ـ تاريخيا (expD) مثل : DATE()

٤ ـ منطقيا (expL) مثل MDATE = DATE()

وقد يكون التعبير صالحا لكل هذه الأنواع وفي هذه الحالة يعبر عنه هكذا :

exp

تذكر...!

تناولنا في هذا الفصل المعلومات التي يجب أن تعرفها عن «كلبر» إذا كنت تستخدمها لأول مرة مثل متطلباتها وحدودها وأنواع ملفاتها وحقولها والعلامات الحسابية والمنطقية التي تستخدمها وسوف تحتاج لكل هذه المعلومات عند تطوير نظم إدارة قواعد بيانات باستخدام «كلبر».

الفصل الثالث

تركيب وتهيئة Clipper

يتنـــاول هذا الفصل كيفية تركيب Clipper

والحصول على معلومات مساعدة وتجهيز كل من:

ملف CONFIG.SYS

ملف AUTOEXEC.BAT

لتوفيق كلبر للعمل مع الحاسب.

تركيب Clipper

لا توجد حماية على برنامج Clipper لذلك فإن تركيبه يتم بسهولة شديدة تتلخص فيما يلي:

١ - أنشيء دليلا فرعيا بالأمر الآتي:

```
C:\> MD\ CLIPPER
```

٢ - انتقل إلى هذا الدليل بالأمر الآتي:

```
C:\> CD\ CLIPPER
```

٣ - ضع قرص النظام في مشغل القرص A ثم انتقل إلى مشغل A بالأمر

```
C:\> A:
```

٤ - أدخل الأمر الآتي:

```
C:\> CLIPCOPY C:
```

وهذا الأمر من شأنه تنفيذ ملف تجميعي (Batch file) بالاسم CLIPCOPY.BAT وهو الذي يتولى نسخ ملفات Clipper إلى القرص الصلب.

الحصول على معلومات مساعدة

يوجد ضمن أقراص برامج Clipper ملف اسمه READ_ME.FIX يشتمل على معلومات مفيدة عن البرنامج وطريقة استخدامه وأسماء الملفات الموجودة على الأقراص الأخرى. اطبع هذا الملف للرجوع إليه عندما تحتاج إلى ذلك.

ملف التهيئة CONFIG.SYS

يشتمل ملف CONFIG.SYS على التعليمات المستخدمة في تهيئة الحاسب مثل عدد الملفات التي يمكن فتحها وحجم المحطات الانتقالية لنقل البيانات (Buffers). ففي كل مرة تدير مفتاح تشغيل الحاسب يبدأ نظام التشغيل DOS في البحث في الدليل الرئيسي في الاسطوانة التي بدأت تشغيل الحاسب منها عن ملف اسمه CONFIG.SYS. فإذا وجد هذا الملف فإنه يقرأه ويبدأ في تنفيذ التعليمات الموجودة

بداخله. أي أن هذا الملف يعدل أو يغير من المسار التلقائي لنظام التشغيل. أما إذا كان هذا الملف غير موجود فإن نظام التشغيل يخصص قيما تلقائية للأوامر التي تهيىء الحاسب.

ولأن صانعي الحاسبات يضعون في حسابهم أن تكون متوافقة مع البرامج بصفة عامة فيلزمك تهيئة الحاسب ليتوافق مع Clipper بصفة خاصة ولذلك لابد من وجود ملف CONFIG.SYS على الدليل الرئيسي بالقرص الثابت ليشتمل على العدد المناسب للملفات التي يجب أن تخصص بواسطة الحاسب.

تحديد عدد الملفات والمحطات الانتقالية

عادة تخصص حاسبات اي. بي. ام والمتوافقة معها عدد ٨ ملفات لتفتح معا في وقت واحد ويحتاج نظام التشغيل DOS إلى خمسة منها. معنى ذلك أن عدد الملفات التي تبقى لتفتح معا مع قاعدة البيانات هو: ٨-٥=٣ ملفات. ولأن Clipper يستطيع التعامل مع عدد من الملفات يصل إلى ٢٥٥ ملفا. فإذا حاولت تشغيل قاعدة البيانات بدون ملف CONFIG.SYS فإن الحاسب سيخصص لك العدد التلقائي وهو ٨ ملفات. منها خمسة لنظام التشغيل. ولذلك فإن العدد الباقي وهو ثلاثة ملفات لن يكون كافيا.

وكانت إصدارات 2.0 فما دون من نظام التشغيل تسمح بفتح حد أقصى من الملفات قدره عشرون ملفا وابتداءا من الاصدار 3.3 زاد هذا العدد إلى ٢٥٥ ملفا ولذلك يجب أن تراعي عدد الملفات التي ستضعها في ملف CONFIG.SYS تبعا للاصدار الذي تعمل به من نظام التشغيل وينصح مصممو «كلبر» بإنشاء ملف -CON FIG.SYS أو تعديله بحيث يشتمل على الأمرين التاليين:

```
FILES  20
BUFFERS  8
```

حيث إن الأمر الأول يخصص عددا من الملفات مقداره عشرون لتفتح معا (خمسه منها لنظام التشغيل وخمسة عشر يمكن فتحها مع «كلبر».

والأمر الثاني يخصص عددا من المحطات الانتقالية (buffers) التي يحجزها نظام التشغيل في ذاكرة الحاسب مقداره ثمانية (وهذا العدد هو الذي تنصح به الشركة المنتجة لقاعدة البيانات «كلبر»).

ويمكن تعريف هذه المحطات الانتقالية (buffers) بأنها مساحة داخل ذاكرة الحاسب تستخدم مؤقتا لمعالجة البيانات (المدخلات والمخرجات) وعادة تحجز في شكل مساحات متجاورة (blocks). وعادة يتحدد عدد المحطات (buffers) التي يمكن تخصيصها في ذاكرة الحاسب بحجم الذاكرة المتاحة في الحاسب المستخدم والحد الأقصى للعدد المسموح هو 99.

وفيما يلي سنوضح كيفية إنشاء أو تعديل ملف CONFIG.SYS إذا لم تكن تعرف كيفية إنشائه أو تعديله لتضمينه الأوامر التي شرحناها.

ملف CONFIG.SYS ملف مكتوب بشفرة ASCII ولذلك فيمكن كتابته أو تعديله بأي محرر للسطور (Editor) أو منسق للنصوص (Word processor) معروف لك أو حتى بمحرر السطور الموجود ضمن ملفات DOS والمسمى EDLIN بحيث يشتمل على السطرين التاليين:

```
FILES=20
BUFFERS=8
```

فإذا كنت لا تعرف كيف تنشىء ملفا نصيا وتريد أن تضع هذا الملف على الدليل الرئيسي على القرص الثابت فيجب اتباع الخطوات الآتية:

١ - انتقل إلى الدليل الرئيسي باستخدام أمر \CD.

٢ - عندما يظهر أمامك محث نظام التشغيل <\:C اكتب الأمر الآتي:

```
C:\> COPY CON: CONFIG.SYS
```

٣ - اكتب السطرين التاليين:

```
FILES=20
BUFFERS=8
```

٤ - اضغط مفتاح F6 وعندما تظهر لك هذه العلامة Z^ اضغط مفتاح الادخال.

٥ ـ ستظهر لك الرسالة الآية:

1 File(s) copied

ومعناها أن الملف تم إنشاؤه.

تحديد ملف ANSI.SYS

ملف ANSI.SYS أحد الملفات التي تأتي مع DOS ومهمته تهيئة بعض البرامج (software) والأجهزة (hardware) لتعمل مع الحاسب.

فإذا كانت التطبيقات المعدة بواسطة «كلبر» تستلزم وجود هذا الملف فيجب أن يشتمل ملف CONFIG.SYS على السطر الآتي:

DEVICE=ANSI.SYS

ومن ناحية أخرى إذا كانت برامجك ستنفذ على حاسبات غير متوافقة مثل WANG أو تستلزم شاشة ANSI فيجب أن تضيف ملف ANSI.OBJ الذي يأتي ضمن حزمة برامج Clipper إلى الملفات التي سيتم ربطها لاستخراج ملف EXE.

ملاحظة: ستعرف كيفية ترجمة وربط البرامج في الفصل السادس إن شاء الله.

ويجب الانتباه إلى أن إضافة ملف ANSI.OBJ إلى البرنامج/البرامج المراد ربطها (linking) سيتسبب في غياب مؤشر الشاشة ولذلك فإننا ننصح بعدم إضافة هذا الملف إلا في حالة الضرورة القصوى التي تتطلب تشغيل البرامج على حاسبات غير متوافقة.

تجهيز ملف Autoexec.bat

تحتاج برامج وتطبيقات «كلبر» أثناء التنفيذ إلى التعامل مع الذاكرة لتخزين حقول الذاكرة (Memory variables) أولا استخدام المحطات الانتقالية (buffers) لأغراض الفهرسة أو لاستدعاء برامج أخرى نتيجة لتنفيذ أمر RUN أو لمعالجة البيانات وإجراء العمليات الحسابية والمنطقية اللازمة.

ويتم إجراء هذه العمليات في معظم البرامج تلقائيا إلا أنك قد تحتاج لاجراء تحكم أو توجيه أكثر لاستخدام برامجك لذاكرة الحاسب أو لتقسيم ذاكرة الحاسب. لتحقيق أقصى استفـادة من حجم الـذاكـرة الموجودة عندك وفي هذه الحالة يجب استخدام أمر SET وهو أحد أوامر نظام التشغيل لتعديل الطريقة التلقائية التي يتم بها عمل نظام التشغيل. ولأن هذا الأمر له تأثير على استخدام وإدارة ذاكرة الحاسب فيجب وضعه في ملف Autoexec.bat ليتم تنفيذه تلقائيا في كل مرة يدار فيها مفتاح تشغيل الحاسب وملف Autoexec.bat ينفذ تلقائيا في كل مرة تدير فيها مفتاح تشغيل الحاسب ويبدأ نظام التشغيل في العمل. فبمجرد انتهاء الحاسب من قراءة ملف COFIG.SYS يبحث عن هذا الملف فإذا وجـده نفذ جميع التعليمات التي يشتمل عليهـا وملف Autoexec.bat ملف يكتب بشفرة ASCII ولذلك نستطيع إنشاءه أو تعديله بأي محرر للكلمات أو منسق للنصوص.

والشكل العام لأمر نظام التشغيل SET كما يلي:

SET CLIPPER=[Vnnn][;Rnnn][;nnn][;Xnnn][;Fnnn]

(لا توجد مسافة قبل أو بعد علامة = الموجودة في الأمر)

وقبل شرح المعطيات الواردة في هذا الأمر نود الاشارة إلى أن الأرقام التي تتبعها محسوبة بالكيلو بايت فمثلا إذا استبدلت nnn في أحد المعطيات بالرقم 128 فمعنى هذا 128K

ومهمة هذا الأمر تعريف تطبيقات «كلبر» بإمكانيات الحاسب وكيفية استغلال الذاكرة.

أما إذا أدخلت أمر SET بدون اسم أو معطيات هكذا

SET

فستحصل على شكل مثل هذا الشكل

COMSPEC=C:\COMMAND.COM

PROMPT=PG

PATH=C:\DOS;C:\CLIPPER,C:\UTILITY

ونوضح فيما يلي الأوامر الواردة بهذا الشكل:

* COMSPEC: يشير إلى اسم الدليل الذي سيجد فيه DOS الملف COMMAND.COM وعادة يوضع ملف COMMAND.COM على الدليل الرئيسي في حالة التشغيل من قرص ثابت أو على القرص المرن في حالة التشغيل من قرص مرن.

* PG: لاظهار اسم الدليل الحالي متبوعا بعلامة >

* PATH: يحدد لنظام التشغيل أين يبحث عن الملفات أو الأوامر التي يتم إدخالها من تحت نظام التشغيل.

والآن نعود لشرح أمر SET والاختيارات التي يمكن إدخالها إليه.

SET CLIPPER=[Vnnn][;Rnnn][;Ennn][;Xnnn][;Fnnn]

وجود هذه الأقواس [ ] معناه أن ما بداخلها اختياري فيجوز إضافته إلى الأمر كما يجوز الاستغناء عنه حسب حاجتك وتؤثر المعطيات الواردة في الأمر على طريقة تنفيذ تطبيقات «كلبر» على النحو التالي:

* الاختيار V

تخصص «كلبر» مساحة الذاكرة المحددة بعد الاختيار V للمتغيرات التي يتم وضعها بالذاكرة والتي يطلق عليها Memory variables فإذا لم يتم تحديد هذا الاختيار فإن كلبر تخصص ٢٠٪ من مساحة الذاكرة لهذا الغرض بحد أقصى ٤٤ك.ب وتسمح «كلبر» بتخصيص حد أقصى من حقول الذاكرة قدره ٢٠٤٨ ويحتاج كل حقل داخل الذاكرة إلى ٢٢ بايت لتسجيل اسمه ونوعه وطوله. فإذا كانت برامجك مثلا تحتاج في حدود ٢٥٦ حقلا فقط فمن المناسب تخصيص ٦ كيلوبايت لهذا الاختيار هكذا:

"V006" وقد تم حساب ٦ ك.ب طبقا للمعالة التالية:.

V=X * 22/1024

V=256 * 22/1024 = 5.5K (6K)

ويكون الأمر المناسب في هذه الحالة هو

SET CLIPPER=V006

وبهذا تستطيع توفير مساحة قدرها ٤٤-٦=٣٨ك.ب لاستغلالها لمعالجة البيانات أو لأغراض الفهرسة.

وأيضا إذا حددت مساحة أكثر من المطلوبة فمثلا: V=036 في هذا المثال تعني أن ٣٠ك.ب من ذاكرة الحاسب تبقى غير مستغلة.

ملاحظة: لا يمكن تشغيل حقول الذاكرة على الذاكرة الاضافية للحاسب (Extended memory)

*** الاختيار R**

الرقم الذي يتبع الاختيار R يخصص مساحة من ذاكرة الحاسب بالكيلوبايت (K) تستخدم لأغراض الفهرسة (Index buffers) ولتنفيذ البرامج الخارجية التي يتم استدعاؤها بأمر RUN

ملاحظة: كلمة Index buffer تختلف عن كلمة Buffer التي شرحناها في ملف CONFIG.SYS

والمساحة المخصصة من الذاكرة لكل من البرامج التي يتم استدعاؤها من خارج النظام بأمر RUN وعمليات الفهرسة واحدة. ولذلك فإذا خصصت لهذا الاختيار مساحة أكثر من المتاحة عندك فستحصل على رسالة خطأ مفادها أن مساحة الذاكرة غير كافية.

وإذا كنت تستخدم ذاكرة إضافية (Extended Memory) فإن الجزء المتاح من الذاكرة الرئيسية RAM يخصص لاستدعاء البرامج الخارجية أي لتحميل نسخة ثانية من ملف COMMAND.COM ويتم إجراء عمليات الفهرسة على الذاكرة الاضافية إلا إذا كنت خصصت مساحة قدرها صفر للذاكرة الاضافية بالاختيار E الذي سنشرحه فيما بعد.

* الاختيار E

يحدد هذا الاختيار أقصى مساحة من الذاكرة الاضافية (Expanded Memory) بالكيلوبايت يمكن استخدامها بواسطة تطبيقات «كلبر». فإذا أغفلت هذا الاختيار فسيفهم كلبر أنه يجوز له استخدام كل المساحة المتاحة من الذاكرة الاضافية بحد أقصى قدره ١ ميجابايت.

وتستخدم «كلبر» الذاكرة الاضـافية لأغراض الفهرسة فقط. والحد الأدنى من المساحة التي يجب تخصيصها بواسطة هذا الاختيار هو ١٦ ك.ب.

* الاختيار X

يستخـدم هذا الاختيار لاستبعـاد مساحة من الذاكرة بالكيلوبايت حتى لا تستخدمها تطبيقات أو برامج «كلبر».

ويستخدم هذا الاختيار في الغالب لأغراض الاختبار فيمكنك تشغيل برامجك على مساحة أقل من المتاحة في الذاكرة لتقرير أقل مساحة يتطلبها النظام. فإذا كان حاسبك يشتمل على ذاكرة قدرها ٦٤٠ك.ب وخصصت للاختيار X مساحة قدرها ٢٥٦ك.ب. فمعنى هذا أنك تسقط مساحة قدرها ٢٥٦ك.ب من المساحة الكلية للذاكرة وتتعامل مع مساحة قدرها ٣٨٤ك.ب فقط. فإذا أمكنك تشغيل النظام بعد استبعاد ٢٥٦ك.ب من الذاكرة فيمكنك توثيق النظام على أنه يحتاج المساحة قدرها ٣٨٤ك.ب من الذاكرة.

وهنـاك استثنـاء وحيد من هذه القـاعـدة وهـو الـبرامـج الخارجية التي يتم استدعاؤها بأمر RUN لأن مثل هذه البرامج تتعامل مع كل مساحة الذاكرة ولا تعتبر بالمساحة المخصصة للاختيار X

* الاختيار F

يستخدم هذا الاختيار لتحديد أقصى عدد من الملفات يمكن فتحها معا. وقد أوضحنا قبل قليل عند الكلام عن ملف CONFIG.SYS أن النظام يعطينا تلقائيا ٨ ملفات ويسمح لنا بتغيير هذا العدد بما لا يزيد عن ٢٠ في إصدارات ما قبل 3.3

وبما لا يزيد عن ٢٥٥ ملفا في إصدارات 3.3 أو أكثر. فبفرض أن ملف CONFIG.SYS يشتمل على عدد من الملفات قدره ٢٠ ملفا فهذا يعني أن العدد المتاح لجميع التطبيقات هو ٢٠ فإذا أردت تحديد هذا العدد في تطبيقات «كلبر» بخمسة عشر ملفا استخدم هذا الاختيار هكذا:

```
SET CLIPPER=F15
```

في هذه الحالة لن تتعامل كلبر مع أكثر من ١٥ ملفا رغم أن ملف CONFIG.SYS يسمح بالتعامل مع ٢٠ ملفا.

مـثـال:

```
SET CLIPPER=V10;R16;E256;F15
```

ويعني هذا المثال:

- تخصيص مساحة قدرها ١٠ك.ب للمتغيرات التي ستوضع في الذاكرة (Memory Variables) أثناء تنفيذ البرامج.
- تخصيص مساحة قدرها ١٦ك.ب لتعمليات الفهرسة ولتنفيذ البرامج التي يتم استدعاؤها من خارج النظام.
- تخصيص مساحة قدرها ٢٥٦ك.ب من الذاكرة الاضافية لاجراء عمليات الفهرسة بها (الحد الأدنى الذي يمكن تخصيصه للاختيار E هو ١٦).
- تخصيص عدد من الملفات لتفتح معا داخل النظام قدره ١٥ ملفا (منها ٥ لنظام التشغيل).

تـذكـر...!

شرحنا في هذا الفصل كيف تتعامل «كلبر» مع نظام التشغيل DOS . وذلك لتعرف أثناء تصميم نظم إدارة قواعد البيانات كيف يمكن تحقيق أقصى استفادة من الحاسب ومكوناته (مثل الذاكرة الرئيسية والذاكرة الاضافية). وأيضا لتستطيع تشغيل العدد المناسب من الملفات التي تتطلبها برامجك.

الفصل الرابع

ترجمة برامج dBASE III PLUS باستخدام Clipper

يوضح هذا الفصل ما يجب أن تعرفه قبل ترجمة البرامج التي كتبتها باستخدام قاعدة البيانات dBASE III PLUS. ويشتمل على الأوامر والوظائف التي لا تتعامل معها Clipper. والأوامر والوظائف الجديدة في Clipper والتي لم تكن موجودة في dBASE III PLUS. ومن خلال المقارنة بين أوامر كل من البرنامجين والبدائل الموجودة في كل منهما يسهل عليك إعداد برامجك الجديدة أو تعديل برامجك القديمة طبقا للفروق الموجودة بينهما.

توجـد اختـلافـات بسيطة بين برنامج dBASE III PLUS وبرنامج Clipper. ولذلك يجب تعديل برامجك التي كتبتها في الماضي باستخدام قاعدة البيانات dBASE III PLUS قبل ترجمتها باستخدام Clipper. ويتحتم تعديل البرامج لتتوافق مع مترجم «كلبر» في إحدى حالتين:

١) إذا كانت برامجـك تشتمل على أحد الأوامر التي لا تتعامل معها «كلبر» ومعظمها من الأوامر التي تستدعي شاشة كاملة والتي يطلق عليها بلغة «دي بيس» Full Screen Commands. وهذه الأوامر لا تستخدم داخل البرامج إلا نادرا. وإنما يغلب استخدامها في التعامل المباشر مع الحاسب مثل أمر ASSIST أو أمر CREATE أو أمر EDIT أو أمر BROWSE... الخ. وسنورد بعد قليل بيانا تفصيليا بجميع هذه الأوامر. فإذا اشتمل البرنامج المطلوب ترجمته على مثل هذه الأوامر يجب تعديله لحذف هذه الأوامر منه.

٢) عندما تحتاج لاستخدام أوامر أو وظائف غير موجودة في قاعدة البيانات dBASE III PLUS. وليس لها بديل بها مثـل أوامر المصفوفات والقوائم والوظائف الخاصة. وتلجأ إلى استخدام مثل هذه الأوامر والوظائف للاستفادة من التسهيلات التي تقدمها والتي تزيد من قوة البرامج والتطبيقات. وسنورد بعد قليل شرحا مختصرا لجميع هذه الأوامر والوظائف لأننا سنشرحها بالتفصيل في الباب الثالث إن شاء الله.

ويجب الانتباه إلى أن البرامج التي يتم تعديلها بإضافة أحد الأوامر أو الوظائف الخاصة بـ«كلبر» والتي لا توجد ضمن أوامر ووظائف dBASE III PLUS لن تستطيع تشغيلها باستخدام dBASE III PLUS ما لم يتم إلغاؤها مرة ثانية.

أوامر ووظائف «دي بيس ثري بلاس» التي لا تتعامل معها «كلبر»

معظم أوامر ووظائف dBASE III PLUS التي لا تتعامل معها Clipper إن لم تكن كلها ـ نادرة الاستخدام داخل البرامج. وإنما تستخدم من محث dBASE III PLUS للتعامل المباشر مثل أمر APPEND أو EDIT أو CREATE/MODIFY. وهذه

الأوامر تتيح التعامل مع قاعدة البيانات بالاضافة أو الحذف أو التغيير بدون حاجة لكتابة برامج . وهي تصلح أساسا للمبتدئين وغير ذوي الخبرة بالبرمجة. ولأن «كلبر» مصممة أساسا لترجمة وتنفيذ برامج مكتوبة فلا معنى لوجود هذه الأوامر التي يفترض أنها لن تستخدم داخل البرامج ، ومن ناحية أخرى فإن «كلبر» لا تشتمل على نقطة توجيه الأوامر (dot-prompt) التي تتيح التعامل المباشرمعها. ومن ناحية ثالثة فإن «كلبر» لا يستخدمها إلا المبرمجون الذين لديهم برامج يريدون ترجمتها وربطها مع نظام التشغيل لتنفيذها بعد ذلك بصفة مستقلة.

ونود أن نطمئن المبرمجين الذين يستخدمون أحد أو بعض هذه الأوامر إلى أن «كلبر» تؤدي معظم الوظائف التي تؤديها أغلب هذه الأوامر مثل إنشاء وتعديل الملفات والتقارير والملصقات. وإظهار محتويات الملفات وفهرستها عن طريق برامج معدة وجاهزة وموجودة في مكتبتها. وسنعرفها بالتفصيل إن شاء الله في الفصل الثامن ونورد فيما يلي هذه البرامج واستخدامها ـ راجع الفصل الثامن لمزيد من المعلومات عن هذه البرامج ـ.

■ برنامج DBU.EXE (اختصار لعبارة DataBase Utility) يسمح بإنشاء وتعديل ملفات قاعدة البيانات وملفات الفهرسة وبإظهار محتويات الملفات بدون الحاجة إلى كتابة برامج مستقلة أو استخدام مفسر. وهذا البرنامج لا يأتي مباشرة مع حزمة برامج «كلبر» وإنما ينتج من ترجمة وربط الملفات التالية معا نتيجة لتنفيذ ملف MAKEDBU.BAT الذي يأتي ضمن حزمة «كلبر».

DBU.PRG
DBUCOPY.PRG
DBUEDIT.PRG
DBUHELP.PRG
DBUINDEX.PRG
DBUSTRU.PRG
DBUUTIL.PRG
DBUVIEW.PRG

ملاحظة: ستعرف في الفصل السادس إن شاء الله كيف يمكن ترجمة (compiling) وربط (linking) البرامج معا

فإذا أردت الحصول على ملف DBU.EXE نفذ الملف التجميعي MAKEDBU.BAT. وهو يفترض أن كلبر والمكتبات الخاصة به تم تركيبها تحت دليل اسمه CLIPPER\. ثم نفذ هذا الملف فيما بعد بكتابة اسمه هكذا

C:\CLIPPER> DBU

■ برنامج RL.EXE (اختصار لعبارة Report Label) يسمح بإنشاء أو تعديل ملفات التقارير والملصقات. وهذا البرنامج ينتج من ترجمة وربط الملفات الثلاثة التالية معا والتي تأتي ضمن حزمة «كلبر» نتيجة لتنفيذ ملف MAKERL.BAT

RLBACK.PRG

RLDIALOG.PRG

RLFRONT.PRG

■ برنامج LINE.EXE يستخدم لاظهار البرامج على الشاشة أو طباعتها على الطابعة مع ترقيم أوامر البرنامج.

■ برنامج MAKE.EXE. يستخدم هذا البرنامج في حالة التطبيقات الكبيرة التي تشتمل على أكثر من برنامج ليوضح لمترجم «كلبر» البرامج التي طرأ عليها تعديل فقط والتي تحتاج لاعادة ترجمتها وربطها مع نظام التشغيل بدلا من إعادة ترجمة وربط جميع البرامج التي يحتوي عليها النظام.

■ «يستخدم «كلبر» أيضا وظائف جديدة بديلا لبعض الوظائف الموجودة في «دي بيس ثري بلاس» مثل الوظيفة MEMOEDIT() والوظيفة DBEDIT() وهي تسمح بتعديل محتويات الملفات بديلا لأمر BROWSE وسنشرحها بالتفصيل في الباب الثالث إن شاء الله.

ويوضح الجدول التالي الأوامر التي لا يتعامل معها «كلبر» والتي يجب أن تخلو منها برامجك التي كتبتها باستخدام «دي بيس ثري بلاس» التي تنوي ترجمتها وربطها مع نظام التشغيل باستخدام «كلبر».

وأمام كل منها الأمر أو الأوامر البديلة الموجودة في برنامج «كلبر» ويجب أن تكون هذه الأوامر معروفة لديك من خبرتك السابقة بقاعدة البيانات «دي بيس ثلاي بلاس». فإذا كانت كلها أو بعضها غير معروفة لديك فننصحك بالرجوع إلى كتابنا المرجع الشامل لقاعدة البيانات dBASE III PLUS

الأمر في dBASE III PLUS	وظيفته	الأمر المقابل أو البديل في Clipper
APPEND	إضافة سجل أو سجلات جديدة في نهاية الملف المفتوح باستخدام شاشة إضافة السجلات	@...SAY...GET
ASSIST	استدعاء شاشة المساعدة	استخدم برنامج DBU.EXE
BROWSE	إظهار أو تعديل أو إضافة سجلات من خلال نافذة تشتمل حتى ١٧ سجلا في الشاشة الواحدة	DBEDIT()
CHANGE	تعديل سجل/سجلات باستخدام شاشة كاملة لكل سجل وهو يشبه أمر EDIT	لا يوجد. استخدم برنامج DBU
CLEAR FIELDS	يلغي تأثير أمر SET FIELDS TO	لا يوجد
CREATE/MODIFY LABEL	إنشاء أو تعديل ملف الملصقات وهو يظهر شاشة تصميم الملصقة	برنامج RL.EXE

الأمـــــر في dBASE III PLUS	وظيفتـــــــــــــــه	الأمر المقابل أو البديل في Clipper
CREATE/MODIFY REPORT	إنشاء أو تعديل ملف تقارير وهو يظهر شاشة تصميم التقرير	برنامج RL.EXE
CREATE/MODIFY QUERY	إنشاء أو تعديل شاشة تستخدم في الاستفسارات	DBFILTER()
CREATE/MODIFY SCREEN	إنشـاء أو تعديل شاشة لتناسب أغراض الاضافة أو التعديل على الملف الأصلي لقاعدة البيانات	@...SAY...GET
CREATE/MODIFY VIEW	ربط ملفين طبقا لبيانات حقل مشترك بينهما ليصبحا ملفا واحدا	@...SAY...GET
DISPLAY/LIST FILES	إظهـار ملفـات قاعـدة البيانات الموجودة على الدليل الحالي	DIR
DISPLAY/LIST MEMORY	إظهـار محتـويات الذاكرة (اسم الحقل ونوعه وحجمه)	@...SAY
DISPLAY/LIST STATUS	إظهـــار معلومـات عن الملفـات المختارة وملفات الفهرسة وأرقام المناطق العاملة وحالة أوامر SET ووظيفة كل مفتاح من مفاتيح الوظائف	لا يوجد. استخدم أوامر مختلفة
DISPLAY/LIST STRUCTURE	إظهار مواصفات ملف قاعدة البيانات	AFIELD()
DISPLAY/LIST USERS	يستـخـدم في حالـة شبـكـة الاتـصـالات المحلية لمعـرفـة مستخدمي البرنامج	لا يوجد.

الأمـــــر في dBASE III PLUS	وظيفتـــــــــه	الأمر المقابل أو البديل في Clipper
EDIT	تعـديل سجـل/سـجـلات باستخدام شاشة كاملة لكل سجل	لا يوجد. استخدم برنامج DBU
ERROR()	تستخدم هذه الوظيفة مع أمر ON ERROR لتتبع أخطاء البرنامج	لا يوجد
EXPORT TO	يحول ملف قاعـدة البيانـات المفتوح إلى ملف من نوع PFS	لا يوجـد. استخـدم إحـدى الـوظـائـف التي تتيح تحويل الملف إلى شكل آخر.
HELP	إظهـار شاشـة المسـاعدة تعطي معلومـات عن قاعـدة البيانـات والأوامر والوظائف التي تستخدمها	لا يوجـد. صمـم شاشـة لمعلومات المساعدة خاصة بك
IMPORT FROM	يحول ملفـات برنـامج PFS إلى ملفات قاعدة البيانات	لا يوجـد. استخـدم إحـدى الـوظـائـف التي تتيح تحويل الملف إلى شكل آخر.
INSERT	إقحام سجل جديد داخل ملف قاعدة البيانات	لا يوجد. استخدم أمر APPEND BLANK ثم أدخل البيانات
LIST HISTORY	إظهار آخر أوامر تم إدخالها من نقطة توجيه الأوامر	لا يوجد. لأن كلبر لا توجد به نقطة توجيه الأوامر
LOAD	ينقل ملف جاهز للتنفيذ (.BIN) في الذاكرة تمهيدا لتنفيذه فيما بعد بواسطة أمر CALL	لا يوجد. اكتب البرامج اللازمة بلغة التجميع أو سي ثم اربطها مع النظام كله

الأمر في dBASE III PLUS	وظيفته	الأمر المقابل أو البديل في Clipper
LOGOUT	يستخدم في حالة شبكة الاتصالات المحلية للتحكم في حماية الملفات	لا يوجد. لأنك تستطيع وضع كلمة سر داخل برنامج ثم ترجمة البرنامج لحمايته
MESSAGE()	تستخدم في حالة شبكة الاتصالات المحلية لتتبع رسائل الخطأ	لا يوجد
MODIFY COMMAND	استدعاء منسق الكلمات لكتابة أو تعديل برنامج مكتوب بشفرة ASCII	MEMOEDIT()
MODIFY STRUCTURE	تعديل مواصفات ملف قاعدة البيانات	لا يوجد. استخدم أمر CREATE مع وظيفة AFIELD() لكتابة برنامج يقوم بهذه المهمة
ON ERROR/ESCAPE/KEY	يستخدم لأغراض تعقب واكتشاف الأخطاء	وظيفة INKEY() أو أمر SET KEY
RESUME	يستخدم لأغراض تعقب واكتشاف الأخطاء	لا يوجد. استخدم مكتشف الأخطاء الموجود في كلبر
RETRY	يستخدم لاعادة محاولة قراءة سجل غير مسموح	لا يوجد. استخدم طريقة أخرى مثل إظهار رسالة لمشغل البرنامج ليعيد المحاولة. أو عودة البرنامج إلى القائمة الرئيسية
RETURN TO MASTER	ينهي البرنامج الحالي وينقل التنفيذ إلى البرنامج الرئيسي في النظام	لا يوجد. اكتب برنامجا صغيرا يستدعي القائمة الرئيسية للنظام بدلا منه.

الأمـــــر في dBASE III PLUS	وظيفتـــــــــــــه	الأمر المقابل أو البديل في Clipper
SET	يستدعي شاشة كاملة تتيح أداء بعض الوظائف مباشرة مثل تغيير الألوان والدوال ومشغل الأقراص. . . الخ	لا يوجد. اكتب الأوامر التي تقوم بهذه المهام داخل البرنامج
SET CARRY ON/OFF	يسمح أو يمنع نسخ السجل على الشاشة من حالة إدخال سجلات جديدة بأمر APPEND	لا يوجد اكتب برنامجا صغيرا لهذه المهمة
SET CATALOG TO	إنشاء ملف كتالوج (CAT.) أو فتحه أو إغلاقه	لا يوجد. لأن كلبر لا يستخدم ملف الكتالوج لأنه عديم الجدوى داخل البرامج
SET COLOR ON/OFF	يستخدم للتبديل بين الشاشة الملونة وأحادية اللون في حالة اتصال الحاسب بأكثر من شاشة	SET COLOR TO - SETCOLOR() ISCOLOR()
SET DEBUG ON/OFF	تتبع أخطاء البرنامج أثناء تنفيذه	لا يوجد. استخدم مكتشف الأخطاء (Debugger) الموجود في كلبر
SET ECHO	لاختيار إظهار أوامر البرنامج على على الشاشة أثناء تنفيذه أو إلغاء ذلـــك.	لا يوجد. استخدم مكتشف الأخطاء
SET DOHISTORY	لاختيار حفظ الأوامر التي توجه إلى قاعدة البيانات من نقطة توجيه الأوامر أو عدم حفظها	لا يوجد. لأن كلبر لا يشتمل على نقطة توجيه الأوامر
SET FIELDS TO	يحدد الحقول المختارة من الملف التي ستستخدم مع أوامر قاعدة	لا يوجد. لأنه غير ضروري. اذكر الحقول التي تريد إظهارها بعد

الأمر في dBASE III PLUS	وظيفته	الأمر المقابل أو البديل في Clipper
	البيانات	الأمر مباشرة
SET HEADING	لاظهار أو إخفاء أسماء الحقول. التي تظهر مع بعض الأوامر مثل LIST - SUM..	لا يوجد لأن كلبر لا يظهر أسماء الحقول مع أمر LIST أو DISPLAY استخدم الأمر SAY...@ بدلا منه
SET HELP	يحدد هل تريد إظهار رسالة للحصول على مساعدة في حالة حدوث خطأ أم لا .	لا يوجد. لأن كلبر لا يستخدم شاشة المساعدة
SET HISTORY TO	يحدد الأوامر التي سيتم حفظها والتي تدخل من نقطة توجيه الأوامر (العدد المخصص تلقائيا ٢٠ أمرا)	لا يوجد. لأن كلبر لا يستخدم نقطة توجيه الأوامر
SET MEMOWIDTH TO	يحدد عدد الحروف التي ستظهر في السطر الواحد من حقل الملاحظات (العدد المخصص تلقائيا ٥٠ حرفا)	لا يوجد. استخدم البديل وظيفة MEMOEDIT()
SET MENUS ON/OFF	يسمح أو يمنع إظهار شاشة المساعدة التي تظهر في أعلى الشاشة مع أوامر إظهار الشاشة الكاملة	لا يوجد. لأن كلبر لا يستخدم الأوامر التي تستدعي شاشة كاملة
SET SAFETY ON/OFF	يسمح أو يمنع إظهار رسالة تحذيرية عندما تريد الكتابة على ملف موجود من قبل	لا يوجد. لأن كلبر يتعامل مع البرامج فقط. ولا سبيل لادخال أمر خطأ من نقطة توجيه الأوامر

الأمـــــر في dBASE III PLUS	وظيفتـــــــــــــــه	الأمر المقابل أو البديل في Clipper
SET STATUS ON/OFF	يسمح أو يمنع إظهار سطر الحالة (status bar) في أسفل الشاشة	لا يوجد. لأن كلبر لا يظهر سطر الحالة
SET TALK ON/OFF	يسمح أو يمنع إظهار نتائج أو أمر قاعدة البيانات مباشرة على الشاشة	لا يوجد. لأن كلبر لا يتعامل مباشرة مع الشاشة
SET TITLE ON/OFF	يستخدم في حالة استخدام ملف كتالوج لاظهار أسماء الملفات	لا يوجد. لأن كلبر لا يتعامل مع ملف الكتالوج
SET VIEW TO	يفتح ملفا موجودا يشتمل على بيانات ملفين تم ربطهما معا	لا يوجد. لأن كلبر لا يستخدم هذا الملف (.VUE)
SET STEP ON/OFF	يستخدم لأغراض اكتشاف وتعقب الأخطاء	لا يوجد. لأن كلبر به البديل وهو استخدام مكتشف الأخطاء

الأوامر التي يستخدمها «كلبر» وغير موجودة في «دي بيس ثري بلاس»

يوضح الجدول التالي باختصار الأوامر التي تستخدمها Clipper والتي لا توجد في dBASE III PLUS. وقد عالجت dBASE IV معظم أو كل هذه الأوامر المتقدمة التي تميز بها Clipper عن dBASE III PLUS. ويشتمل الباب الثالث على الشرح التفصيلي لهذه الأوامر موضحا بالأمثلة المناسبة. ويمكنك الرجوع إليه للحصول على معلومات تفصيلية عن هذه الأوامر.

الأمـر	وظيفتـــــه
@...BOX	يستخدم لرسم إطار على الشاشة واختيار رمز أو حرف لكل ضلع من أضلاعه أو لكل ركن.
@...PROMPT	يستخدم لكتابة اختيارات قائمة اختيارات
@...SAY...GET...	يسمح بقبول قيمة وإدخالها إلى الملف أو تخزينها بذاكرة إذا توفرت فيها شروط معينة
DECLARE	إنشاء مصفوفة ذات بعد واحد
EXTERNAL	يعرف رموز تدل على أسماء بعض البرامج والاجراءات إذا لم تكن موجودة ضمن النظام ليتعرف عليها برنامج الربط (Linker)
FOR...NEXT	ينشىء دوارة تنفذ عددا محددا من المرات
KEYBORAD	يمسح المحطات الانتقالية (Typeahead Buffer) ويضع بها عبارة حرفية
MENU TO	يستدعي قائمة اختيارات تستخدم الشريط المضاء للانتقال بين اختياراتها بناء على مجموعة أوامر @...PROMPT

الأمــر	وظيفتــــــــــــه
RESTORE SCREEN	يسترجع محتويات الشاشة التي تم تخزينها بأمر SAVE SCREEN
SAVE SCREEN	يخزن محتـويات الشاشة الموجودة داخل حقل ذاكرة أو داخل ذاكرة الحاسب لاسترجاعها فيما بعد
SET CURSOR	يظهر أو يلغي ظهور مؤشر الشاشة
SET KEY	يخصص إجراء معينا (برنامجا صغيرا) لينفذ عند الضغط على أحد المفاتيح أثناء توقف البرنامج مؤقتا نتيجة لأحد أوامر: - WAIT - READ - MENUE TO - INPUT ACCEPT
SET SOFTSEEK	يحدد المكان الذي سيوضع عنده المؤشر داخل الملف إذا لم تجد «كلبر» السجل الذي تبحث عنه. فإذا كان في وضـع ON فإن المؤشر لن يوضـع في نهاية الملف وإنما سيوضع عند أقرب مكان للسجل المطلوب البحث عنه
SET WRAP	تسمـح بانتقـال المؤشر إلى أول اختيار داخـل قائمـة الاختيارات إذا وصل إلى آخر اختيار

الوظائف التي يستخدمها «كلبر» وغير موجودة في dBASE III PLUS

يوضح الجدول التالي باختصار الوظائف التي تستخدمها Clipperوغير موجودة في dBASE III PLUS ويشتمل الباب الثالث على الشرح التفصيلي لهذه الوظائف. ويمكنك الرجوع إليه لمزيد من المعلومات عن هذه الوظائف.

الوظيفة	استخدامها
ACHOICE()	تظهر قائمة تستخدم الشريط المضاء للانتقال بين اختياراتها وتضع اختيارات هذه القائمة داخل مصفوفة
ACOPY()	تنسخ محتويات مصفوفة إلى مصفوفة أخرى
ADEL()	تحذف أحد عناصر المصفوفة
ADIR()	تعطي رقما يوضح عدد الملفات الموجودة على الدليل الحالي ويمكن بالاضافة إلى ذلك نقل معلومات عن هذه الملفات مثل حجمها وتاريخ ووقت إنشائها إلى مصفوفات أخرى.
AFIELDS()	تعطي معلومات عن حقول الملف الحالي داخل مصفوفات
AFILL()	تملأ عناصر مصفوفة بقيمة معينة
AINS()	تضع عنصرا بين عناصر مصفوفة
ALIAS()	تعطي اسما يدل على اسم الملف الموجود في المنطقة المختارة
ALLTRIM()	تحذف الفراغات الموجودة على يمين ويسار عبارة حرفية
ALTD()	تسمح باستدعاء مكتشف الأخطاء أو تلغي إمكانية استدعائه

الوظيفة	استخدامهـــــــــــا
ASCAN()	تبحث عن أول عنصر داخل مصفوفة يتطابق مع عبارة حرفية
ASORT()	ترتب عناصر مصفوفة ترتيبا تصاعديا
BROWSE()	تشبه أمر BROWSE() الموجود في «دي بيس ثري بلاس»
CURDIR()	لاظهار أو لمعرفة اسم الدليل الحالي
DBEDIT()	تسمح بإظهار وتعديل السجل بطريقة مشابهة لأمر BROWSE الموجود في «دي بيس ثري بلاس»
DBFILTER()	تعطي العبارة المستخدمة مع أمر SET FILTER TO
DBRELATION()	تعطي اسم الحقل المتخذ أساسا لربط ملفات قواعد البيانات باستخدام أمر SET RELATION
DBRSELECT()	تعطي رقم المنطقة التي يوجد بها الملف المرتبط مع الملف الحالي بأمر SET RELATION
DTOS()	تظهر التاريخ في شكل عبارة تأخذ هذا الشكل yyymmdd
EMPTY()	تعطي القيمة المنطقية .T. إذا كانت عبارة حرفية لا تشتمل على شيء.
FCLOSE()	تغلق ملف نصي (DOS File)
FCOUNT()	تعطي عدد سجلات الملف المفتوح
FCREATE()	تنشيء ملف نصي (DOS File)
FERROR()	تستخدم لمعرفة نوع الخطأ الذي حدث من استخدام وظائف الملفات
FIELD()	تظهر اسم حقل موجود بالملف
FILE()	لمعرفة هل هناك ملف موجود على الدليل الحالي أم لا
FLOCK()	لغلق الملف الحالي حتى لا يستخدمه الآخرون داخل شبكة الاتصالات

الوظيفة	استخدامهـــــــــا
FOPEN()	لفتح ملف نصي (.TXT)
FREAD()	لقراءة ملف نصي داخل الذاكرة
FREADSTR()	لقراءة جزء من ملف مفتوح
FSEEK()	لوضع المؤشر في مكان ما داخل ملف نصي فتح بإحدى الوظيفتين FOPEN() أو FCREATE()
FWRITE()	لكتابة عدد من الحروف موجودة في حقل ذاكرة داخل ملف نصي
HEADER	تحسب عدد الحروف التي تستخدم كعناوين للحقول في الملف المفتوح
INDEXEXT	تحدد هل تم ربط ملف NDX.OBJ مع النظام المستخدم أم لا
INDEX KEY()	تحدد الحقل المستخدم كمفتاح عند فهرسة الملف
INDEXORD()	تحدد اسم الفهرس الرئيسي الذي يتحكم في ترتيب الملف المفتوح
LASTKEY()	تعطي الشفرة المقابلة لآخر حرف تم الضغط عليه من لوحة المفاتيح
LOCK()	لغلق الملف في حالة استخدام شبكة اتصالات محلية
MEMOEDIT()	تفتح نافذة تشبه أمر BROWSE في «دي بيس» لاظهار أو تعديل حقل الملاحظات أو عبارة حرفية
MEMOLINE()	تسمح بإظهار (أو طباعة) سطر موجود في عبارة حرفية أو حقل ملاحظات
MEMOREAD()	تقرأ ملف نصي (ASCII) من القرص الممغنط
MEMORY()	تحسب المساحة المتبقية من الذاكرة

الوظيفة	استخدامهـــــــا
MEMOWRITE()	تنقل محتويات حقل ملاحظات أو عبارة حرفية إلى ملف نصي على القرص الممغنط
MLCOUNT()	تحسب عدد السطور الموجودة في حقل الملاحظات أو في عبارة حرفية
NETERR()	تستخدم لمعرفة هل نجح أمر USE أو أمر APPEND BLANK أم لا
NETNAME()	تعطي اسم المحطة المتصلة بشبكة الاتصالات المستخدمة
PCOUNT()	تحسب عدد المعطيات (Paramters) التي دخلت إلى البرنامج
PROCLINE()	تحدد رقم السطر الحالي في الاجراء الذي ينفذ
PROCNAME()	تحدد اسم الاجراء الذي ينفذ
RAT()	تبحث عن آخر حرف أو مجموعة حروف موجودة داخل عبارة حرفية
READVAR()	تحدد اسم الحقل الذي يُستدعى بأمر GET أو MENU
RESTSCREEN()	تسترجع جزءا من شاشة حفظت بالوظيفة SAVESCREEB()
RLOCK()	تغلق سجلا في شبكة الاتصالات حتى لا يستخدمه الآخرون
SAVESCREEN()	تحفظ جزءا من الشاشة داخل الذاكرة لاسترجاعه فيما بعد
SECONDS()	تعطي رقما يقابل الوقت الموجود بالحاسب بالثانية
SELECT()	تحدد اسم المنطقة المختارة
SETPRC()	تحدد قيمة لكل من الوظيفة PROW() و PCOL()

الوظيفة	استخدامهـــــــــــا
SOUNDEX()	تحول عبارة حرفية إلى أربعة حروف تستخدم لتعامل العبارة القريبة منها في النطق معاملة هذه العبارة عند البحث عنها أو فهرستها
STRTRAN()	تقوم بالبحث عن عبارة حرفية داخل أخرى واستبدالها بقيمة جديدة
TONE()	تسمح بسماع صوت الجرس بنغمات مختلفة
UPDATED()	تحدد هل تم تعديل حقل أثناء استدعاء أمر READ أم لا

تـذكـر...!

يعتبر هذا الفصل مرجعا سريعا ـ حتى بعد الانتهاء من قراءة الكتاب وفهمه ـ يوضح الأوامر التي يجب أن تخلو منها البرامج التي أعدت في الماضي باستخدام «دي بيس ثري بلاس» قبل تنفيذها بواسطة «كلبر» ويوضح أيضا البدائل المتاحة في «كلبر» لكل منها. ويوضح الفصل أيضا لمحة سريعة عن الأوامر والوظائف التي لا تتعامل معها «دي بيس ثري بلاس»، ومنها تتضح الامكانيات الهائلة التي توفرها «كلبر» في إعداد نظم إدارة قواعد البيانات فإذا أردت معرفة هذه الأوامر والوظائف بالتفصيل فيمكنك الرجوع إلى الباب الرابع ـ مرجع الأوامر والوظائف.

الفصل الخامس
البرمجة باستخدام
«كلبر»

لو اقتصرت المزايا الجديدة في «كلبر» على ترجمة برامجك وربطها مع نظام التشغيل وتشغيلها بعد ذلك أو بيعها في الأسواق بدون حاجة إلى قاعدة البيانات لكان هذا سببا كافيا لشرائها واقتنائها إلا أنها اشتملت على العديد من المزايا الأخرى والتي جاءت في الاصدار اللاحق من dBASE وهو dBASE IV ويوضح هذا الفصل الملامح الجديدة في «كلبر» والتسهيلات التي يجب أن يعرفها مستخدمو «كلبر» لأول مرة.

بالـرغم من أن كلا من «كلبر» و«دي بيس» يستخدم لنفس الغرض تقريبا وبـالرغم من أن معظم الأوامر في البرنامجين واحدة إلا أن هناك فروقا جوهرية في طريقة عمل كل منهما.

تتلخص طريقة عمل dBASE III PLUS في تحميل البرنامج/البرامج المطلوبة للتنفيذ من القـرص إلى ذاكـرة الحاسب بمجرد استدعائها للتنفيذ. ويتولى المفسر مراجعـة الأوامـر أثنـاء التنفيذ أمـرا أمـرا ليتـأكد من خلوها من الأخطاء اللغوية والنحـوية. ويتم إعـادة الملفـات والـبرامج مرة أخرى إلى القرص المغنط بمجرد إغلاقها من داخل البرنامج.

بينما تتلخص طريقة عمل Clipper في تجميع جميع البرامج والملفات التي يتكون منها النظام وترجمتها وربطها مع نظام التشغيل ووضعها في ملف جديد بعد الترجمة يمكن تشغيله منفردا بدون حاجة لتشغيل «كلبر». ويتم تحميل هذا البرنامج الجديد في ذاكرة الحاسب مرة واحدة بمجرد استدعائه للتنفيذ. ولهذا السبب فإن تنفيذ هذه البرامج يتم بسرعة عالية.

وبهذا يتبين أن «كلبر» تتميز على «دي بيس» بالمزايا الآتية:

■ تشتمل على مجموعة من الأوامر والوظائف الجديدة التي تزيد من كفاءتها وتضيف قوة إلى نظم إدارة قواعد البيانات التي يتم إعدادها باستخدامها مثل استخدام المصفوفات والوظائف الخاصة والقوائم المنسدلة و. . . وغيرها. وهي التسهيلات التي جاءت في dBASE IV بعد ذلك. وسبق أن أوضحنا أن البرامج التي سبق إعـدادها باستخدام dBASE III PLUS يمكن ترجمتها باستخدام Clipper بعد التأكد من خلوها من الأوامر التي لا يتعامل معها «كلبر». والتي أوضحناها في الفصل الرابع.

■ تحقيق السرية المطلوبة لحماية البرامج التي تعتبر حقا مكتسبا لأصحابها وأفكارا خاصة بهم. لأن «كلبر» لا تشتمل على نقطة توجيه الأوامر (dot-prompt) التي تتيح لأي فرد تغيير البرامج المصدرية أو الاطلاع عليها.

- يوفر عليك استخدام «كلبر» الانتقال إلى الاصدار الأخير من «دي بيس» وهو dBASE IV للاستفـادة من الأوامـر والـوظائف الجديدة التي تتيح التعامل مع القوائم المنسدلة والمصفوفات وإعداد الوظائف الخاصة. وغيرها من الامكانيات المتقدمة.

- الاستفادة من خبراتك السابقة في البرمجة بلغات أخرى مثل C أو Assembly. لأنه يتيح لك ربط هذه البرامج مع برامج «كلبر» ثم ترجمتها جميعا وربطها مع نظام التشغيل.

إلا أن هذه المزايا والتحسينات لا تقلل من شأن dBASE III PLUS أو dBASE IV (الآن) لأن لكل منهما استخداما خاصة ومزايا عديدة خصوصا للمبتدئين وحديثي العهد بالبرمجة ومما يؤكد ذلك أن الطلب على dBASE III PLUS لم يتوقف أو يقل حتى بعد صدور Clipper. ويمكن اعتبار dBASE III PLUS هي البداية بما تتيحه من التعامل المباشر من خلال نقطة توجيه الأوامر و Clipper امتداد لهذه البداية خصوصا للمتمرسين والمتخصصين في إعداد نظم إدارة قواعد البيانات.

ونـوضـح فيما يلي بعض المـلامح أو التسهيلات الجديدة الموجودة في «كلبر» لتحقيق الفائدة لمستخدمي «كلبر» لأول مرة أو الذين يودون تطوير برامجهم باستخدام «كلبر». وفي الباب الثاني سنوضح باقي الامكانيات والمفاهيم المتقدمة الموجودة في «كلبر» مثل استخدام المصفوفات والتعامل مع أخطاء البرامج واستخدام مكتشف الأخطاء لتعقب وتصحيح أخطاء البرامج.

الوظائف الخـاصـة

(User Defined Functions)

تسمح Clipper بأن تخصص لنفسك وظائف خاصة بك. وتسمى الوظائف الخـاصـة User Defined Functions وتختصر هكـذا UDF وهـذه الـوظائف يمكن استخـدامهـا بنفس الطريقة التي تستخدم بها وظائف «كلبر» كما يمكن دمجها مع إحدى وظائف «كلبر». وتعتبر الوظائف الخاصة من أهم التسهيلات الجديدة التي أضافتها «كلبر» والتي لم تكن موجودة في «دي بيس ثري بلاس».

إنشاء الوظائف الخاصة

يمكن القول إن الوظائف الخاصة وظائف مثلها مثل وظائف «كلبر» المعروفة (راجـع الباب الثالث) والفرق بينهما أن الوظائف الخاصة ينشئها المبرمج لتؤدي له وظيفة أو عملا معينا يتكرر باستمرار داخل النظام الذي يطوره ولا تستطيع وظائف «كلبر» أداء هذا العمل. مثل ذلك البحث عن سجل داخل الملف أو تخصيص قيمة تلقائية لأحد المتغيرات أو تصحيح البيانات الداخلة إلى الملف باستخدام أمر GET...@ ويستخدم أمر FUNCTION لانشاء الوظيفة الخاصة ويأخذ هذا الشكل:

FUNCTION<procedure name>

حيث <procedure name> هو الاسم المختار للوظيفة ويجب أن تنتهي بالأمر:

RETURN <exp>

ويشترط في الاسم المختار للوظيفة (<procedure name>) ألا يتشابه مع وظائف أو أوامر Clipper المعروفة.

وبمجـرد إنشـاء وظيفة خاصة بك يمكنك استخدامها في أي مكان داخل البرنامج لتنفيذ بعض الأوامر لأنها تقوم مقام برنامج فرعي أو مجموعة من الأوامر (Submodules) يتم استدعاؤها للتنفيذ داخل نفس البرنامج أو من برنامج أو إجراء آخر.

استخدام الوظائف الخاصة

ليس من الضروري أن تضع الـوظيفة الخاصة داخل ملف برنامج (.prg) مستقل. فقد توضع في بداية البرنامج كما قد توضع في نهاية البرنامج . أو قد توضع بين ملفات الاجراءات (procedure files) أو وظائف خاصة أخرى. وعند استدعاء إحدى الوظائف الخاصة للتنفيذ فإنها تستقبل بيانات من خارج مجموعة الأوامر التي تشتمل عليها لتحل محل المعطيات (parameters) الموجودة بها. وتقوم Clipper بتنفيذ الأوامـر التي تشتمـل عليهـا الوظيفة باستخدام البيانات الداخلة إليها وفي النهاية تخصص .قيمة للوظيفة الخاصة وهي ما يطلق عليها <exp> وهذا هو السبب أن أمر RETURN يجب أن يبدو هكذا :

RETURN <exp>

وسوف يتضح ذلك من خلال الأمثلة التالية التي توضح أهم استخدامات الوظائف الخاصة . وسوف نتحاشى في هذه الأمثلة الأوامر الجديدة في Clipper لأننا لم نشرحها بعد .

المثال التالي يستخدم الوظيفة الخاصة SALFUN لحساب صافي الراتب بعد خصم الضرائب بالمعادلة الآتية :

صافي الرابت = الراتب الأساسي - (الراتب × معدل الضريبة)

ويتم إدخال الراتب الأساسي ومعدل الضريبة من خلال البرنامج . أما الراتب الصافي فيتم حسابه من خلال الوظيفة الخاصة وبالتالي إرساله إلى البرنامج . ويوضح شكل ١ـ٥ الجزء من البرنامج الذي يقبل الراتب الأساسي ومعدل الضريبة ويستخدم الوظيفة الخاصة SALFUNC() لحساب معاملة صافي الراتب

```
INPUT "Enter base salary ==> " TO M_SALE
INPUT "Enter tax rate =====> " TO M_TAX
? "Net salary: " + STR(SALFUN(M_SALE,M_TAX))
```

شكل ١ـ٥

وإذا راجعت وظائف «كلبر» فلن تجد بينها الوظيفة ()SALFUN ولذلك لابد من إنشاء هذه الوظيفة بأمر FUNCTION لتؤدي الوظيفة المطلوبة وهي حساب صافي الراتب. ويوضح شكل ٢ـ٥ الأوامر التي تشتمل عليها هذه الوظيفة الخاصة.

```
FUNCTION SALFUN          && SALFUN : اسم الوظيفة
PARAMETERS SAL,TAX       &&         اقبل الراتب ومعدل الضريبة
                         && حتى 128 متغير يمكن قبولها من خارج الوظيفة
NET = SAL - (SAL*TAX)    &&    NET ضع الراتب الصافي في حقل الذاكرة
RETURN NET
```

شكل ٢ـ٥

أما النتيجة التي سنحصل عليها أثناء تنفيذ البرنامج فتبدو كما يلي:

```
Enter base salary ===> 4200
Enter tax rate ======> .07
Net salary: 3906
```

ويفترض أن المشغل أدخل الرقم 4200 ردا على الرسالة الأولى والرقم 07. ردا على الرسالة الثانية.

يوضح شكل ٣ـ٥ وظيفة خاصة تستخدم لضبط عبارة وسط السطر بعد حذف البيانات الموجودة على يمينها أو يسارها وتعتمد على أن العبارة (string) وطول السطر (length) ستدخل من البرنامج عند استدعائها.

```
FUNCTION CENTER                    && CENTER  اسم الوظيفة:
  PARAMETER STRING,LENGTH          && اقبل متغيرين من خارج الوظيفة
  PRIVATE LSPACE,RSPACE,CLC_VAL && اذا وردت اسماء هذه المتغيرات
                                   && في برامج اخرى يجب الا تؤثر القيم
                                   && التي تخصص لها على محتويات هذه
                                   && المتغيرات في هذه الوظيفة
  * الاوامر التالية تحسب الفراغات على يمين ويسار العبارة وبالتالي مكانها *
  LSPACE = INT((LENGTH - LEN(STRING))/2)
  RSPACE = LENGTH - LSPACE - LEN(STRING)
  CLC_VAL = SPACE(LSPACE) + STRING + SPACE(RSPACE)
RETURN  CLC_VAL
```

شكل ٣ـ٥

ونتناول بالشرح الأوامر التي وردت في هذه الوظيفة :

١ ـ الأمر *FUNCTION*

يعلن اسم الوظيفة والاسم المختار هنا هو CENTER وبذلك يفهم «كلبر» أن الأوامر التالية حتى أمر RETURN تخصها.

٢ ـ الأمر *PARAMETERS*

تفترض الوظيفة أن المعطيات التي ستدخل من البرنامج هي العبارة (string) وطول السطر الذي ستكتب عليه هذه العبارة (length).

٣ ـ الأمر *PRIVATE*

يخصص القيم التي ستعطى للمتغيرات الثلاثة المذكورة بعده داخل هذه الوظيفة فقط حتى ولو وردت أسماء هذه المتغيرات في مكان آخر أو في برنامج آخر داخل النظام.

٤ ـ الأمر *LSPACE=*

يطرح طول العبارة (string) من طول السطر الذي ستكتب عليه (length) ثم يقسم الناتج على ٢ ويضع الناتج في حقل ذاكرة اسمه LSPACE. وهو يمثل عدد الفراغات التي ستظهر على يسار العبارة. لاحظ أن الوظيفة INT() تقرب الناتج لأقرب رقم صحيح.

٥ ـ الأمر *RSPACE=*

يطرح ناتج الخطوة رقم ٤ من طول السطر ثم يطرح من الناتج طول العبارة ويضع الناتج في حقل ذاكرة اسمه RSPACE وهو يمثل الفراغات التي ستظهر على يمين العبارة.

٦ ـ الأمر *CLC_VAL*

يضع العبارة التي ستظهر وسط السطر بالاضافة إلى ما قبلها وما بعدها من فراغات ـ حتى لا تظهر أي كتابة كانت موجودة على نفس السطر من قبل ـ يضع كل ذلك في حقل ذاكرة اسمه CLC_VAL

٧ ـ الأمر RETURN

يرسل محتويات حقل الذاكرة المسمى CLC_VAL إلى البرنامج الذي يستدعي الوظيفة.

وعنـدما ترغب في استخدام هذه الوظيفة الخاصة فيجب استخدامها داخل البرنامج أو أي برنامج آخر داخل النظام بالطريقة التالية:

CENTER (<expC>, <expN>)

حيث:

<expC>: هي العبارة التي تريد كتابتها وسط السطر بعد حذف الفراغات الموجودة قبلها وبعدها على نفس السطر.

<expN>: هو عرض السطر الذي تريد أن تبدو العبارة وسطه.

فبفرض أنك تريد أن تظهر اختيارات القائمة بحيث يخصص لكل اختيار مساحة قدرهـا ٢٠ عمـودا ويفـرض أن الاختيار المطلوب إظهاره وسط هذه المساحة بعد استبدال الفراغات الموجودة على يمينه ويساره بفراغات هو كلمة "Adding"

فالأمر المناسب في هذه الحالة هو

```
@ 12,5 PROMPT CENTER("Adding",20)
```

استخدام الإجراءات
(Proceudres File)

الاجراء عبارة عن مجموعة أوامر تؤدي وظيفة معينة أي يمكن اعتباره برنامجا صغيرا يقوم بمهمة محددة داخل برنامج كبير ويتم استدعاؤه للتنفيذ مثل أي برنامج بإصدار أمر

DO <Procedure>

ويمكن قبول معطيات أو قيم من خارج الاجراء بإضافة الاختيار WITH إلى أمر DOS<procedure> ويمكن قبول معطيات (parameters) من خارج الاجراء تصل إلى ١٢٨ ويمكن أن يتعامل الاجراء مع حقول الذاكرة التي سبق إنشاؤها.

والاجراء يمكن أن يوضع داخل ملف مستقل أو كجزء من برنامج بشرط أن يبدأ بكلمة PROCEDURE

ورغم أن عدد الاجراءات التي يمكن وضعها داخل ملف dBASE III PLUS واحد محدود باثنين وثلاثين إجراء فإن عددها مفتوح في Clipper في حدود حجم الذاكرة المتاحة.

مثال:

يوضح شكل ٤ـ٥ إجراء صغيرا يقوم بمهمة حساب صافي الراتب بعد إدخال إجمالي الراتب ومعدل الضرائب. وهي نفس النتيجة التي حصلنا عليها باستخدام

```
PROCEDURE SALARY          && SALARY  اسم الاجراء:
PARAMETERS SAL,TAX        && اقبل الراتب ومعدل الضريبة
                          && حتى 128 متغير يمكن قبولها من خارج الاجراء
NET = SAL - (SAL*TAX)     && NET ضع الراتب الصافي في حقل الذاكرة
? "Net salary: " + LTRIM(STR(NET))
RETURN
```

شكل ٤ ـ ٥

الوظيفة الخاصة ()SALFUN قبل قليل لأن الاجراء والوظيفة الخاصة متشابهان إلى حد كبير.

ويتم تنفيذ هذا الاجراء بالأمر التالي:

```
DO SALARY WITH 14200,.08
```

بفرض أن الراتب ١٤٢٠٠ ومعدل الضريبة ٨٪ ستحصل على النتيجة التالية بعد تنفيذ البرنامج

```
Net salary: 13064
```

ويتم وضع الاجراءات أو الوظائف الخاصة في ملف مستقل أو في نهاية البرنامج أو ضمن ملف إجراءات (procedure file) كما هو الحال في «دي بيس ثري بلاس» وإذا استخدم ملف إجراءات (procedure file) فإنه يتم ربطه مع النظام بأمر SET PROCEDURE إلا أن استخدام أمر SET PROCEDURE في «كلبر» يختلف عنه في «دي بيس» لأن «كلبر» تضع كل البرامج في الذاكرة في بداية العمل ولذلك فلست في حاجة لاغلاق ملف الاجراءات بأمر CLOSE PROCEDURE وبالتالي، فإن هذا الأمر في «كلبر» ليس ذا معنى أما في «دي بيس» فلابد من استخدام أمر CLOSE PROCEDURE لاغلاق الملف.

وبمجرد ترجمة ملف الاجراءات إلى برنامج هدف (obj.) فإنه يصبح جزءا من النظام. وبالتالي فليس هناك حاجة لفتحه أو إغلاقه في كل مرة.

وإن شاء الله سيكون لنا عودة لشرح استخدام الوظائف الخاصة وملف الاجراءات داخل النظام عند شرح التطبيقات الشاملة باستخدام «كلبر» في الباب الرابع من هذا الكتاب.

مثال شامل:

انظر البرنامج الموجود في شكل ٥-٥ وهو نموذج يوضح كيفية وضع الاجراءات والوظائف الخاصة داخل البرنامج. ويشتمل هذا البرنامج على أمر واحد

```
* Program    : NETSAL.PRG
* Programmer: Magdi M. Abu Al-Ata
* Note ,     : يوضح هذا البرنامج كيف تستخدم الاجراء او الوظيفة
*              الخاصة داخل البرنامج
DO SALARY WITH 14200,.08
*
*------------------------------------------------------*
* هذا الاجراء  يحسب الراتب الصافي                        *
*------------------------------------------------------*
PROCEDURE SALARY          && SALARY  اسم الاجراء:
PARAMETERS SAL,TAX        && اقبل الراتب ومعدل الضريبة
                          && حتى 128 متغير  يمكن قبولها من خارج الاجراء
NET = SAL - (SAL*TAX)     && NET ضع الراتب  الصافي في حقل الذاكرة
YOURNET = "Net salary: " + LTRIM(STR(NET))
@ 12,5 SAY CENTER(YOURNET,50)
RETURN
*
*------------------------------------------------------*
* هذه الوظيفة الخاصة تضع العبارة وسط السطر بعد حذف      *
* الفراغات التي قد تظهر على يمينها او يسارها             *
*------------------------------------------------------*
FUNCTION CENTER                     && CENTER  اسم الوظيفة:
  PARAMETER STRING,LENGTH           && اقبل متغيرين من خارج الوظيفة
  PRIVATE LSPACE,RSPACE,CLC_VAL     && اذا وردت اسماء هذه المتغيرات
                                    && في برامج اخرى يجب الا  تؤثر القيم
                                    && التي تحصل‌ها على محتويات هذه
                                    && المتغيرات في هذه الوظيفة
  * الاوامر التالية  تحسب الفراغات على يمين ويسار العبارة وبالتالي مكانها
  LSPACE = INT((LENGTH - LEN(STRING))/2)
  RSPACE = LENGTH - LSPACE - LEN(STRING)
  CLC_VAL = SPACE(LSPACE) + STRING + SPACE(RSPACE)
RETURN  CLC_VAL

* End Of Program: NETSAL.PRG
```

شكل ٥-٥

فقط يستدعي الاجراء الموجود داخل البرنامج للتنفيذ مستخدما متغيرين هما 14200 وتمثل الراتب، 08. وتمثل معدل الضريبة.

والاجراء بدوره يستخدم الوظيفة الخاصة الموجودة داخل البرنامج .

ويمكن وضع كل من الاجراء SALARY والوظيفة الخاصة CENTER داخل ملف إجراءات مستقل يسمى procedure file وفي هذه الحالة يجب أن يشتمل برنامج NETSAL.PRG على الأمر التالي :

```
SET PROCEDURE TO <procedure file name>
```

وتلاحظ في البرنامج NETSAL.PRG أننا استخدمنا الوظيفة الخاصة CENTER التي شرحناها في البند السابق كما هي بدون تعديل. أما الاجراء SALARY فقد استبدلنا الأمر:

```
?"Net salary:"+LTRIM(STR(NET))
```

بالأمرين التاليين :

الأول

```
YOURNET="Net salary:"+LTRIM(STR(NET))
```

لنضع التعبير كله داخل حقل ذاكرة جديد اسمه YOURNET (لاحظ أن هذا الحقل سينشأ حرفي) .

الثاني

```
@12,5 SAY CENTER(YOURNET,50)
```

وفي هذا الأمر استخدمنا الوظيفة ()CENTER الموجودة بالبرنامج لضبط محتويات YOURNET وسط المساحة المحددة وهي ٥٠ عامودا بعد حذف أية بيانات موجودة على يمينها أو يسارها .

إعداد القوائم ذات الشريط المضاء
Highlighted Bar Menus

لا شك أن استخدام القوائم التي تعتمد على تحريك الشريط المضاء (Highlighted Bar) لتحديد الاختيار المطلوب أسهل بكثير من تلك التي تعتمد على رقم أو حرف يدل على الاختيار المطلوب كما كنا نفعل في «دي بيس ثري بلاس».

وتعتمد معظم التطبيقات الحديثة على مفهوم استخدام الشريط المضاء لتحديد الاختيار المطلوب داخل قائمة الاختيارات وذلك لسهولة استخدامه بالاضافة إلى أنه يعطي انطباعا مريحا للمستخدم. وتستخدم «كلبر» هذه الطريقة أيضا. ولأن هذه الطريقة لم تكن متاحة بطريقة مباشرة في «دي بيس ثري بلاس» فقد أضافت «كلبر» مجموعة جديدة من الأوامر تسهل استخدام هذا النوع من القوائم هي :

@...PROMPT

SET MESSAGE

SET WRAP

MENU TO

وسنتناول هذه الأوامر بالشرح ثم نقدم مثالا شاملا يستخدم كل هذه الأوامر لبناء قائمة اختيارات.

الأمر @...PROMPT

يستخدم هذا الأمر لاظهار اختيار (prompt) في مكان محدد على الشاشة. فمثلا

@5,10 PROMPT "Enter data"

يظهر عبارة Enter data على الشاشة عند السطر ٥ والعمود رقم ١٠ عندما تستدعي القائمة للتنفيذ بواسطة أمر MENU TO (سنشرح أمر MENU TO بعد قليل).

فإذا أضيف إلى أمر @...PROMPT الاختيار MESSAGE فإن الرسالة المختارة ستظهر في أسفل الشاشة ـ ما لم يتم تحديد سطر آخر ـ عندما يتم وضع

الشريط المضاء فوق هذا الاختيار داخل القائمة. ففي هذا المثال:

```
@5,10 PROMPT "Enter data" MESSAGE;
"Enter new item to the Inventory"
```

تظهر عبارة Enter new item to the inventory في السطر الأخير من الشاشة عندما يتم وضع الشريط المضاء فوق الاختيار Enter data.

الأمر SET MESSAGE

يحدد هذا الأمر رقم السطر الذي ستظهر فيه الرسالة المختارة مع أمر @...PROMPT فمثلا الأمر

```
SET MESSAGE TO 1
```

يتسبب في ظهور الرسالة في السطر رقم ١.

وإضافة الاختيار CENTER إلى الأمر يضع الرسالة وسط السطر. فمثلا الأمر:

```
SET MESSAGE TO 1 CENTER
```

يظهر الرسالة المحددة في وسط السطر رقم ١.

ويجب الانتباه إلى أن هذا الأمر لا يحذف البيانات الموجودة بالسطر قبل إظهار الرسالة. ولذلك فإذا كانت الرسالة أقل من المحتويات الموجودة في السطر قبل إظهارها فإن جزءا من البيانات السابقة سيظهر بجانب الرسالة ولذلك ننصح بحذف المسافات الموجودة قبلها أو بعدها أولا.

ملاحظة: راجع الوظيفة الخاصة CENTER() الموجودة بهذا الفصل تحت عنوان استخدام الوظائف الخاصة. لأنها تقوم بمهمة حذف هذه الفراغات الموجودة يسار أو يمين أي عبارة قبل إظهارها.

الأمر SET WRAP

أمر SET WRAP أمر مفصلي يستخدم في حالتين: نعم (ON) أو لا (OFF) ويعني الوضع ON السماح للشريط المضاء بالانتقال إلى أول إختيار في القائمة عندما يصل إلى آخر اختيار بها باستخدام مفاتيح الأسهم.

الأمر MENU TO

وهو الأمر الخاص باستدعاء القائمة ويأخذ هذا الشكل:

MENU TO <memory variable>

حيث <memory variable> اسم لمتغير موجود بالذاكرة.

عندما يصل البرنامج إلى أمر MENU TO يتم استدعاء أوامر @...PROMPT ويتوقف تنفيذ البرنامج مؤقتا وينتظر المستخدم لضغط أحد المفاتيح التي تتسبب في تنفيذ أحد اختيارات القائمة. ويتم الانتقال من اختيار لآخر داخل القائمة بواسطة مفاتيح الأسهم أو مفاتيح أخرى. أما تنفيذ الاختيار فيتم بوضع الشريط المضاء فوق الاختيار وضغط مفتاح الادخال أو بالضغط على أول حرف من الاختيار المطلوب.

وعندما يتم اختيار واحد من اختيارات القائمة يتم تخزين رقم السطر الذي يدل على هذا الاختيار في حقل ذاكرة فمثلا إذا اخترت الاختيار الثالث من القائمة فيتم تخزين الرقم ٣ في حقل الذاكرة. أما إذا ضغطت مفتاح Esc أثناء ظهور القائمة فسيتم تخزين الرقم صفر في حقل الذاكرة. ويمكن استخدام أمر DO CASE لاستدعاء برنامج معين للتنفيذ بناء على تنفيذ أحد اختيارات القائمة.

ملاحظة: راجع هذا الأمر والأوامر الثلاثة السابقة بالتفصيل في الباب الثالث من هذا الكتاب لتتعرف على هذه الأوامر وعلى وظائف المفاتيح المستخدمة داخل قائمة الاختيارات.

والمثال الموجود في شكل ٦ـ٥ يوضح كيفية استخدام الأوامر الأربعة لرسم اختيارات قائمة وتنفيذها.

```
SET WRAP ON                         && اسمح للمؤشر بالتحرك من اسفل لاعلى
SET MESSGAE TO 1                    && اظهر الرسالة المختارة فى السطر رقم 1
@ 10,05 PROMPT " Maintenance    " MESSAGE " Add, Delete, Edit   "
@ 12,05 PROMPT " Query          " MESSAGE " Ask questions       "
@ 14,05 PROMPT " Reports        " MESSAGE " Reports Menu        "
@ 16,05 PROMPT " Exit           " MESSAGE " Exit to DOS         "
MENU TO ACTION
DO CASE
   CASE ACTION = 1                  && اذا اخترت الإختيار الاول
     DO STMAINT
   CASE ACTION = 2                  && اذا اخترت الإختيار الثانى
     DO STINQ
   CASE ACTION = 3                  && اذا اخترت الإختيار الثالث
     DO STRPT
   CASE ACTION = 4 .OR. ACTION = 0  && اذا اخترت الإختيار الرابع
                                    && او ضغطت مفتاح Esc
     RETURN
ENDCASE
```

شكل ٦-٥

استخدام أمر FOR...NEXT لإنشاء دوارة

بالاضافة لأمر DO WHILE الذي تستخدمه «دي بيس ثري بلاس» لانشاء الدوارة تستخدم «كلبر» أمر FOR...NEXT أيضا لانشاء دوارة (loop) بطريقة مشابهة لاستخدام الأمر في لغة «بيسك».

ويقوم هذا الأمر بتكرار تنفيذ مجموعة من الأوامر الواقعة بين كلمة FOR وكلمة NEXT عددا محددا من المرات. وهو يوفر إنشاء عداد قبل بداية الدوارة وزيادة العداد في كل مرة يتكرر فيها تنفيذ الدوارة وهو ما كنا نفعله باستخدام أمر DO WHILE.

فمثلا المثال التالي يسمح بتكرار الأوامر الواقعة بين DO WHILE... ENDDO ٣ مرات.

```
M_COUNT = 1
DO WHILE M_COUNT <= 3
   + < أوامر >
   M_COUNT = M_COUNT + 1
ENDDO
```

ونفس النتيجة نحصل عليها باستخدام أمر FOR...NEXT كما يلي:

```
FOR M_COUNT = 1 TO 3
  * < أوامر >
NEXT
```

وكما تلاحظ فإن الأوامر المستخدمة في المثال الثاني أقل من نظيرتها في المثال الأول وأسهل في فهمها. ولذلك فإننا ننصح باستخدام أمر FOR...NEXT في حالة الدوارات التي يتم تنفيذها لعدد محدد من المرات لأن «كلبر» تقوم بإنشاء العداد تلقائيا وزيادته في كل مرة يتكرر تنفيذ الدوارة.

ويستخدم أمر FOR...NEXT بكثرة مع المصفوفات.

راجع الأمثلة التي تستخدم أمر FOR...NEXT مع المصفوفات في الفصل السابع والباب الثالث من هذا الكتاب.

إذا اشتمل أمر FOR...NEXT على الاختيار EXIT فإن الدوارة ستغلق قبل الوصول إلى العدد المحدد بالأمر وينتقل التنفيذ إلى الأمر التالي لها. وهو شبيه بالاختيار EXIT داخل أمر DO WHILE...ENDDO.

والمثال الموجود في شكل ٧ـ٥ يوضح كيفية استخدام هذا الأمر.

```
USE STUDENTS
FOR M_COUNT = 1 TO 20
   ?
   ?
   ? LTRIM(FIRSTNAME) + " " + LTRIM(MIDNAME) + "  " + LASTNAME
   ? LTRIM(ADDRESS) + "," + LTRIM(CITYT) + " ."
   SKIP
   IF EOF()
      EXIT
   ENDIF
NEXT
```

شكل ٧ـ٥

التعامل مع ملفات خارجية

تستطيع «كلبر» التعامل مع ملفات تختلف عن مواصفات ملفات قاعدة البيانات DBF. وهي الملفات النصية التي يتعامل معها DOS وتحقق هذه الميزة فائدة كبيرة عند التعامل مع الحقول الحرفية (character) وحقول الملاحظات (memo)

ويتم إنشاء أو تعديل أو صيانة هذه الملفات النصية بواسطة مجموعة جديدة من الوظائف تبدأ جميعها بحرف F. ويوضح الجدول التالي هذه الوظائف باختصار شديد فإذا أردت التعرف عليها بالتفصيل يمكنك الرجوع إلى الباب الثالث من هذا الكتاب.

الوظيفة	استخدامهـــــــــــــا
FCREATE()	لانشاء ملف نصي (text file)
FCLOSE()	لاغلاق ملف نصي
FILE()	لمعرفة هل يوجد ملف معين على الدليل الحالي أم لا
FERROR()	لمعرفة نوع الخطأ الذي حدث من استخدام وظائف الملفات
FOPEN()	لفتح ملف نصي
FREAD()	لقراءة ملف نصي داخل الذاكرة
FREADSTR()	لقراءة جزء من ملف مفتوح
FSEEK()	لوضع المؤشر في مكان ما داخل ملف نصي فتح بإحدى الوظيفتين FOPEN() أو FCREATE()
FWRITE()	لكتابة عدد من الحروف موجودة في حقل ذاكرة في ملف نصي

استخدام التعبيرات بدلا من اختيارات بعض الأوامر

توجد أوامر كثيرة في «كلبر» يمكن استبدال اختياراتها بتعتبير معين فمثلا أمر SET COLOR يمكن أن يكتب هكذا:

```
SET COLOR TO W+/B+,GR+/R+
```

ومعناه أن لون الكتابة أبيض والخلفية زرقاء ولون الكتابة في الشاشة المعكوسة (enhanced) كتلك التي تظهر بعد أمر GET أصفر ولون الخلفية أحمر. كما يمكن أن يكتب بهذا الشكل

```
SET COLOR (COLVAR)
```

وفي الصورة الأخيرة يجب أن توضع الاختيارات المطلوبة لاظهار ألوان الشاشة داخل تعبير حرفي وهو التعبير الذي يذكر اسمه بين هذين القوسين ( )

مـــثـــال:

```
STORE "W+/B+,GR+/R+" TO COLVAR
SET COLOR TO (COLVAR)
```

كما أن كثيرا من أوامر SET...ON/OFF تستخدم تعبيرا منطقيا (<expL>) بدلا من OFF أو ON. انظر الشكل العام للأمر التالي:

```
SET PRINT ON/OFF/(<expL>)
```

فإذا أردنا وضع الطابعة في حالة ON فأمامنا إحدى طريقتين:

الأولى استخدام الأمر بهذا الشكل

```
SET PRINT ON
```

والثانية استخدام الأمر بهذا الشكل

```
STORE .T. TO PRVAR
SET PRINT (PRVAR)
```

وفي الطريقة الثانية فإن القيمة المنطقية .T. تعني ON بينما تعني القيمة المنطقية .F. الوضع OFF

وقد يكون (<expL>) نتيجة مقارنة قيمتين معا. فإذا كانت محتويات (<expL>) نتيجة مقارنة قيمتين هي القيمة المنطقية .F. فإن «كلبر» ستقرأ الأمر هكذا:

SET PRINT OFF

```
=======
* Program: KEY.PRG
SET KEY -9 TO COMPACC        && -9 = F10 Key
CLEAR
DO WHILE .T.
  MCOMP=SPACE(3)
  MACCOUNT=SPACE(8)
  @ 10,5 SAY "Enter company name: " GET MCOMP
  @ 12,5 SAY "Enter account no. : " GET MACCOUNT
  READ                        && READ produces a wait state
  IF MCOMP = " "
     RETURN
  ENDIF
ENDDO
*
* If F10 pressed when the program pausd the following PROCEDURE will
* executes
PROCEDURE COMPACC
PARAMETERS PROG,LINE,VRRBLE && Must accept parameters even if they
                            && are not used.
                            && Parameters here are significant, as the
                            && listing belongs6the pendig GETs
OLDAREA = SELECT()          && Store current workarea in memvar: OLDAREA
SELECT 0                    && Select first unused workarea
SAVE SCREEN                 && Save current screen to a buffer
@ 0,0 CLEAR                 && Don't use CLEAR. It clears pending GETs
DO CASE
   CASE UPPER(VARBLE) = "MCOMP"  && If you nedd inquiring company
     * <commands to list companies>
   CASE UPPER(VARBLE) = "MACCOUNT"  && If you nedd inquiring account
     * <commands to list accounts>
ENDCASE
WAIT "Press a key"   && After listing companies and/or accounts
                     && wait for another keystroke, then
                     && restore saved screen and resume READ command
RESTORE SCREEN
SELECT OLDAREA                  && Return to original workarea
RETURN
```

شكل ٨ـ٥

استخدام مفاتيح الوظائف لتنفيذ برنامج أو إجراء

يمكن استدعاء برنامج أو إجراء (procedure) للتنفيذ عندما يتوقف تنفيذ البرنامج مؤقتا انتظارا لضغط أحد المفاتيح كما يحدث مع الأوامر الآتية:

READ - WAIT - ACCEPT - INPUT - NENU TO

انظر المثال التالي:

SET KEY –9 TO PROCA

في هذا المثال فإن 9- تعني الضغط على مفتاح F10. وبناء على ذلك فإن ضغط مفتاح F10 أثناء توقف تنفيذ البرنامج مؤقتا نتيجة لأحد الأوامر المذكورة يتسبب في استدعاء الاجراء PROCA وتنفيذه.

وفيما يلي نلقي الضوء على أمر SET KEY بشيء من التفصيل.

يشبه أمر SET KEY أمر DO <Procedure> المعروف في أن كليهما يستدعي برنامجا أو إجراء معينا للتنفيذ. إلا أن الاجراء المطلوب يتم تنفيذه فقط أثناء توقف البرنامج مؤقتا بناء على ضغط المفتاح المناسب. وعادة يتوقف تنفيذ البرنامج مؤقتا انتظارا لضغط أحد المفاتيح مع الأوامر التالية:

READ - WAIT - INPUT - ACCEPT - MENU TO

ويمكن تحديد حتى ٣٢ مفتاحا لتنفذ كل منها إجراء معينا أثناء الضغط عليه. وتشمل هذه المفاتيح مفاتيح الوظائف أو دمج مفتاح Alt أو Shift مع مفتاح آخر. باستثناء مفتاح F1 لأنه مخصص دائما لبرنامج المساعدة.

ويجب الانتباه إلى أن تنفيذ أمر SET KEY يتسبب في نقل ٣ قيم إلى الاجراء الذي يتم استدعاؤه وهذه القيم أو المعطيات (parameters) هي: اسم البرنامج المستدعي، ورقم السطر الذي تسبب في استدعاء الاجراء، واسم المتغير (Memory Variable) الذي ينتظر الادخال والموجود بالذاكرة. وهذا الأمر يشبه أمر SET FUNC-TION في أن كليهما يستدعي إجراء معينا نتيجة ضغط أحد المفاتيح إلا أن هذا الأمر

يأخذ أولوية في التنفيذ إذا تعارض مع أمر SET FUNCTION والبرنامج الموجود في «شكل ٨ـ٥» يوضح كيفية استخدام هذا الأمر لاستدعاء إجراء مهمته إظهار أسماء الشركات و/أو أرقام الحسابات في برنامج إدخال بيانات.

المصفوفات

المصفوفات في «كلبر» عبارة عن بيانت تخزن بالذاكرة داخل حقول ذاكرة (Memory variables) ويتم ربط هذه البيانات معا تحت مظلة واحدة باسم واحد. وهو اسم المصفوفة والمصفوفات التي تتعامل معها «كلبر» مصفوفات ذات بعد واحد. وهي بذلك تختلف عن المصفوفات التي تستخدمها لغات «بيسك» أو «كوبول» أو «دي بيس فور» مثلا. لأن المصفوفات التي تستخدمها هذه اللغات مصفوفات ذات بعد واحد أو ذات بعدين.

وتوضع عناصر المصفوفة في سطر واحد فمثلا إذا كنا نرغب في إنشاء جدول يشتمل على أسماء شهور السنة وعلى أسماء ٥ سنوات. فيجب إنشاء مصفوفة باسم معين لتشتمل على ١٢ عنصرا (بعدد شهور السنة) ومصفوفة أخرى تشتمل على ٥ عناصر (عدد السنوات المطلوبة).

وتشتمل المصفوفة الواحدة حتى ٤٠٩٦ عنصرا ويتم التعامل مع كل عنصر موجود داخل المصفوفة كما لو كان حقل ذاكرة مستقل. ولذلك فيمكن أن تشتمل المصفوفة الواحدة على أكثر من نوع بمعنى أن عناصرها يمكن أن يكون بعضها رقميا وبعضها حرفيا وبعضها تاريخيا وبعضها منطقيا.

ولأن إنشاء المصفوفات واستخدامها يبدو غريبا أو صعبا خصوصا للمبرمجين الذين يتعاملون مع المصفوفات داخل براجحهم لأول مرة فقد خصصنا الفصل السابع لشرح المصفوفات وإنشائها وتعبئتها ونسخها وحذفها وترتيب عناصرها والبحث فيها.

تـذكـر...!

اشتمل هذا الفصل على معظم التسهيلات والمفاهيم الجديدة التي تشتمل عليها «كلبر». والتي تعتبر جديدة على مبرمجي «دي بيس ثري بلاس» وهذه التسهيلات والمفاهيم الجديدة هي التي تضفي قوة على نظم إدارة قواعد البيانات التي يتم تطويرها باستخدام «كلبر» وتسهل إعدادها.

وقد خصصنا لهذه المفاهيم فصلا مستقلا برغم أنها سترد في مرجع الأوامر والوظائف في الباب الرابع من هذا الكتاب نظرا لأهميتها. ورغبة في التأكيد عليها وزيادة توضيحها. وتشمل هذه المفاهيم الجديدة ما يلي:

- الوظائف الخاصة
- استخدام الاجراءات
- إعداد قوائم الاختيارات
- إنشاء دوارة بأمر FOR...NEXT
- التعامل مع ملفات خارجية
- استخدام تعبيرات جديدة داخل أوامر «كلبر»
- استخدام مفاتيح الوظائف لتنفيذ إجراء معين.

الفصل السادس

ترجمة البرامج وربطها مع نظام التشغيل

Compiling and Linking

يعتبر هذا الفصل العمود الفقرى لكتابنا هذا فهو يشرح طريقة ترجمة (Compiling) البرامج المصدرية إلى برامج هدف (Object).OBJ وربطها (Linking) مع بعضها لاستخراج ملف جاهز للتنفيذ (.EXE) تحت محت نظام التشغيل DOS ويتعرض لشرح جميع المعطيات التي تتحكم في توجيه عملية الترجمة (Compiling) والربط (Linking) باستخدام Plink86 Plus وأخيرا يشرح البرمجة بطريقة الاحلال (Overlays) واستخدام برنامج MAKE وفي النهاية ستكون قادرا على إنشاء واستخدام الملفات الآتية:

.CLP

.LNK

يتناول هذا الفصل كيفية تحويل برامجك من PRG. وهو البرنامج بشفرة ASCII إلى صورة جاهزة للتنفيذ بواسطة الحاسب يطلق عليها EXE. (Executalbe).

ولأن Clipper لا توجد بها نقطة توجيه أوامر (Dot-prompt) ولا منسق للكلمات (Editor) فيجب أن تختار منسق الكلمات الذي ترتاح له لكتابة البرامج بشفرة ASCII وبالرغم من كثرة البرامج التي تستخدم شفرة ASCII لكتابة النصوص مثل WordStar أو Professional Write . . . أو غيرها. إلا أننا ننصح ببرنامج اسمه SPFPC وهو سهل جدا في تعلمه واستخدامه بالاضافة إلى أنه يناسب كتابة البرامج أكثر من غيره من البرامج الجاهزة لأن كتابة البرامج يناسبها Editor أما الخطابات والرسائل فيناسبها Word Processor.

تحويل البرامج المصدرية إلى برامج جاهزة للتنفيذ

يتم تحويل البرامج المصدرية (source) إلى صورة جاهزة للتنفيذ باستخدام نظام التشغيل باتباع خطوتين أساسيتين هما:

١) ترجمة البرامج المصدرية وتسمى دائما Source code إلى برنامج هدف يسمى دائما Object code.

٢) ربط برامج الهدف التي يتم الحصول عليها في الخطوة السابقة مع بعضها ومع الوظائف التي تتطلبها من مكتبة البرامج للحصول على برنانج جاهز للتنفيذ يسمى دائما Executable form ويختصر هكذا EXE.

وسنناقش كلا من هاتين الخطوتين بالتفصيل فيما يلي.

أولا: ترجمة البرامج

يتولى مترجم «كلبر» تحويل البرامج المصدرية إلى شفرة خاصة يفهمها الحاسب تسمى Object code. وتعرف البرامج المصدرية دائما بالاسم المتد "PRG."(من كلمة Program) بينما تعرف برامج الهدف التي تنتج من ترجمة البرامج المصدرية بالاسم المتد OBJ. (من كلمة Object)

فإذا تم ربط برامج الهدف مع بعضها بواسطة برنامج الربط ويسمى Linker نتج ملف أو برنامج جاهز للتنفيذ. هذا البرنامج في صورته الجاهزة للتنفيذ يسمى EXE. (من كلمة Executalbe).

وتتيح «كلبر» ترجمة البرامج والاجراءات والوظائف الخاصة كل على حدة أو مجتمعة.

والبرامج التي يتم ترجمتها باستخدام مترجم «كلبر» تتحول إلى لغة يفهمها الحاسب فقط ولذلك لا يمكن إرجاعها إلى حالتها السابقة بعد ترجمتها. ولذلك يجب أن تحتفظ بالنسخة الأصلية (المصدرية) للبرنامج إذا كنت تنوي تعديله في المستقبل لأن التعديل يتم على النسخة الأصلية وبالتالي يعاد ترجمة البرنامج بعد تعديله للحصول على نسخة جديدة منه.

وأبسط صور الترجمة هي استدعاء Clipper تحت محث نظام التشغيل متبوعا باسم الملف الرئيسي في النظام المطلوب ترجمته. فبفرض أن البرنامج الرئيسي في النظام الذي نريد ترجمته اسمه INVNTRY.PRG فالأمر اللازم لترجمة النظام هو:

```
CLIPPER INVNTRY
```

ويتم ترجمة البرنامج الرئيسي وجميع البرامج وملفات الاجراءات والوظائف الخاصة المرتبطة بهذا البرنامج. فمثلا إذا كان برنامج INVNTRY.PRG يشتمل على الأمرين التاليين في مكان ما بداخله.

```
DO PRG1
SET PROCEDURE TO INVPROC
```

فإن Clipper سيبحث في الدليل الحالي عن ملف اسمه PRG1.PRG وملف آخر اسمه INVPROC.PRG ليدخلها ضمن برنامج الهدف الذي ينشئه.

أما إذا اشتمل البرنامج المطلوب ترجمته على برنامج غير موجود فإن «كلبر» يظهر الرسالة التالية:

```
CANNOT OPEN, ASSUMED EXTERNAL
```

ومعناها أن كلبر لم يعثر على ملف PRG. بهذا الاسم.

ويمكنك ترجمة أي برنامج PRG. منفردا وربطه مع برنامج أو برامج أخرى بعد ذلك بواسطة برنامج الربط Plink86 plus

التحكم في برنامج الترجمة

ألصيغة التي أوردناها لترجمة البرامج منذ قليل هي أبسط صورة لترجمة البرنامج والحصول على ملف OBJ. إلا أن «كلبر» تتيح استخدام سبعة اختيارات أخرى يمكن إضافة واحدة منها أو أكثر للتحكم في برنامج الترجمة لبعض النتائج التي تريدها بالضبط.

والصورة العامة لأمر الترجمة هي :

CLIPPER <filename> [–l] [–m][–0][–p][–q][–s][–t]

ويجب أن تدخل أحد أو كل هذه الاختيارات بالحروف الصغيرة (Lower Case Letters) فإذا قررت إضافة أكثر من اختيار واحد فيجب ترك فراغ بين كل اختيار والاختيار الذي يليه.

ويوضح الجدول التالي الاختيارات السبعة ومعنى كل منها.

الاختيار	وظيفتـــــــــــــــه
–l	ترجمة البرنامج بدون إضافة رقم السطر إلى برنامج OBJ. والميزة من إضافة هذا الاختيار توفير ٣ حروف من مساحة القرص عن كل أمر في البرنامج. إلا أننا ننصح بعدم استخدام هذا الاختيار ليضيف لك المترجم رقم السطر أمامه ليسهل عليك التعرف عليه في حالة حدوث خطأ ما.
–m	ترجمة ملف واحد فقط. ويستخدم هذا الاختيار بصفة خاصة مع برنامج MAKE ـ سنشرح هذا البرنامج قبل نهاية هذا الفصل ـ وإذا استخدمت هذا الاختيار فإن البرامج أو الاجراءات التي يستدعيها البرنامج لن يتم ترجمتها معه.

الاختيار	استخدامـــــــــــــــــــه
-o	نقل البرنامج المترجم إلى دليل آخر ويمكن الاستفادة من هذا الاختيار في حالة نقل الملف المترجم على قرص مرن مثلا. ويتطلب هذا الاختيار إضافة اسم الدليل الذي تنوي وضع الملف المترجم إليه هكذا: CLIPPER <filename> -o C:\path حيث path اسم الدليل الموجود على القرص.
-p	هذا الاختيار مفيد لأولئك المبرمجين الذين لا يملكون قرصا صلبا لأنه يضع مترجم «كلبر» في ذاكرة الحاسب ثم ينتظر حتى يتم إدخال قرص البرامج في مشغل القرص. إلا أننا ننصح بالاعتماد على قرص صلب في مثل هذه الأحوال.
-q	يمنع ظهور رقم السطر أثناء عملية الترجمة وليس له تأثير على أرقام السطور التي توضع في ملف OBJ.
-s	هذا الاختيار مفيد لأغراض اكتشاف الأخطاء. فهو يتيح قراءة أوامر البرنامج بواسطة المترجم بدون إنشاء ملف OBJ. فإذا اكتشف المترجم أخطاء فسيظهر رسالة خطأ ولن ينشيء ملف OBJ.
-t	يوجه المترجم لانشاء ملف مؤقت على دليل آخر لأن «كلبر» تنشيء ملفا مؤقتا أثناء الترجمة بالاسم الممتد $$$. فإذا أردت زيادة سرعة الترجمة فيمكنك أن تطلب من «كلبر» أن يكتب هذا البرنامج المؤقت على الذاكرة RAM وهذا الاختيار يتطلب الشكل الآتي: CLIPPER filename -t D:\path

ترجمة قائمة برامج باستخدام ملف CLP.

أحيانا نحتاج لترجمة أكثر من ملف للحصول على ملف OBJ. لكل منها تمهيدا لربطها معا بعد ذلك ويستخدم لهذا الغرض ملف يخصص له الاسم الممتد CLP. وهو ملف مكتوب بشفرة ASCII ويشتمل على أسماء الملفات المطلوب ترجمتها. ويتم تجميع الملفات الأصلية (PRG.) كلها بعد الترجمة في ملف OBJ. واحد

فبفرض أن لدينا ملف اسمه MULTI.CLP وأنه يشتمل على البرامج التالية :

PRG1

PRG2

UTY

BAK

(يمكن كتابة أسماء البرامج داخل ملف CLP. بأي منسق للنصوص أو بواسطة محرر النصوص الموجود في DOS) فإذا أردت ترجمة كل هذه البرامج مرة واحدة أدخل الأمر الآتي تحت محث نظام التشغيل DOS.

```
CLIPPER @MULTI
```

وفي هذا الأمر فإن علامة @ تشير إلى أن هذا الملف من نوع CLP. وأنه يشتمل على أسماء البرامج المطلوب ترجمتها وليست البرامج نفسها. في هذه الحالة سيتم ترجمة البرامج الأربعة في ملف OBJ. واحد.

والميزة من استخدام ملف CLP. لتجميع أسماء الملفات أنك تعرف تماما ما هي الملفات التي استخدمت في البرنامج المترجم وأنها توفر كتابة أسماء البرامج مرة ثانية إذا احتجت لاعادة الترجمة لأي سبب.

ثانيا: ربط البرامج التي سبق ترجمتها (OBJ.) معا

. الخطوة الثانية لتحويل البرامج المصدرية إلى صورة جاهزة للتنفيذ هي ربط البرامج المترجمة (OBJ.) مع مكتبة «كلبر» وأي مكتبة أخرى قد تلزم لتحويل البرامج إلى صورة جاهزة للتنفيذ (EXE.) .

وتشتمل «كلبر» على مكتبتين هما:

١ ـ CLIPPER.LIB

٢ ـ EXTEND.LIB

وتشتمل كلتاهما على جميع أوامر ووظائف «كلبر» ـ راجع شرح الأوامر والوظائف بالتفصيل في الباب الثالث من الكتاب.

وبرنامج الربط الذي يقوم بهذه المهمة هو Plink86 plus ويجب أن يعلم برنامج الربط أسماء المكتبة أو المكتبات التي تشتمل على الأوامر والوظائف الموجودة في البرامج. وعادة يبحث برنامج البربط Plink86 plus في مكتبة كلبر CLIPPER.LIB تلقائيا. فإذا كانت بعض الأوامر أو الوظائف موجودة في مكتبة أخرى مثل مكتبة EXTEND.LIB فيجب أن يشار إليها عند استخدام برنامج Plink86 plus.

ويجوز أن تضع بعض الوظائف الخاصة داخل مكتبة ثالثة وفي هذه الحالة يجب أن تشير إلى اسم هذه المكتبة عند استخدام برنامج Plink86 plus بالاضافة إلى اسم مكتبات «كلبر». لأن من مزايا «كلبر» أنه لا يتوقف عند الأوامر والوظائف الواردة بكتاب الشركة المنتجة والذي يباع مع حزمة «كلبر» بل يتعادها إلى أبعد من ذلك ولذلك يطلقون عليه عبارة Open architecture أي البرنامج ذو البناء المفتوح بحيث يستطيع أي شخص أن يضيف إلى هذا البناء حسب إمكانياته.

ولـذلك فيمكنك بناء مكتبة خاصة بك تضع فيها الوظائف التي تؤدي لك أعمالا معقدة ومتكررة إذا كنت خبيرا بلغات البرمجة مثل C أو Assembly كما يمكنك استخدام واحدة من المكتبات الجاهزة التي تباع في الأسواق ونود التنبيه بهذه المناسبة إلى أنه توجد بالأسواق العديد من المكتبات الجاهزة التي تشتمل على مئات بل آلاف الـوظائف التي يمكن ربطها مع «كلبر» والتي تساعد في توفير وقت المبرمج والقيام بالكثير من الوظائف المعقدة التي لا يستغنى عنها أحد مثل البحث عن سجل معين أو إعداد الرسوم البيانية أو تلك الخاصة بشبكات الاتصالات أو الخاصة بالوظائف الحسابية والمالية والاحصائية المعقدة. . . وغيرها كثير. وسنشير إلى أهم هذه المكتبات وعناوين الشركات المنتجة لكل منها في الملحق الثالث بإذن الله.

تشغيل برنامج الربط Plink86 plus

يمتاز برنامج Plink86 plus بأنه الوحيد الذي يستخدم مفهوما يطلق عليه بلغة الحاسب Overlay ومعناها الاحلال (سنشرح هذا المفهوم بالتفصيل قبل نهاية هذا الفصل).

ويتلخص هذا المفهوم باختصار في تقسيم البرامج الكبيرة والتي تحتاج أثناء تشغيلها إلى مساحة أكبر من مساحة الذاكرة إلى أجزاء. ويتم وضع الجزء الرئيسي من البرنامج الذي لا يمكنه الاستغناء عنه في ذاكرة الحاسب بصفة دائمة لأنه عادة يستدعي باقي الأجزاء أما باقي أجزاء البرنامج فإنها توضع على القرص الممغنط ويتم وضع الجزء المطلوب للتنفيذ فقط داخل ذاكرة الحاسب بينما تبقى باقي الأجزاء على القرص الممغنط فإذا أردنا التعامل مع جزء البرنامج الموجود على القرص فإن جزء البرنامج الذي ينفذ يرجع إلى مكانه على القرص ويحل محله الجزء الجديد.

والعيب الوحيد في برنامج Plink86 plus أنه أبطأ من برنامج الربط IBM PC Linker V2.40 الموجود في DOS فإذا لم تكن في حاجة لاستخدام مفهوم الاحلال وتعرف كيف تستخدم برنامج Link من خلال DOS فيمكنك استخدام برنامج الربط IBM PC V2.40 لأنه أسرع. ولأن برنامج الربط الموجود في DOS ليس موضوع هذا الكتاب فسنقتصر على شرح كيفية استخدام برنامج Plink86 plus

توجد ٤ طرق لتشغيل برنامج الربط Plink86 plus

١) إدخال الاختيارات اللازمة لبرنامج الربط Plink86 plus من محث نظام التشغيل في سطر واحد.

٢) استخدام ملف تجميعي (Batch file) لتضع فيه الأمر/الأوامر التي تتحكم في برنامج الربط.

٣) استخدام الطريقة التبادلية (Interactive mode) لاتمام عملية الربط.

٤) تشغيل برنامج الربط Plink86 plus من خلال ملف LNK. وهو يشبه ملف CLP. الذي شرحناه عند الكلام عن الترجمة.

وفي كل هذه الـطرق توجد مجموعة من الاختيارات التي تتحكم في برنامج الـربط Plink86 plus. وسنتناول فيما يلي هذه الاختيارات ومعنى كل منها قبل أن نشرح تشغيل برنـامـج Plink86 plus بالطرق الأربعة سالفة الذكر. وقد تبدو هذه الاختيارات غامضة أو غير مفهومة خصوصا لمن يستخدم Plink86 plus لأول مرة لكن بعد عرض بعد الأمثلة لاستخدامها سينجلي هذا الغموض.

الاختيار	استخدامـــــــــــــــه
#	تستخدم هذه العلامة إذا كنت ستستخدم Plink86 plus من خلال ملف LNK. لتوضح أن هذا السطر عبارة عن تعليقات أو ملاحظات أما في حالة إدخال الاختيارات من محث نظام التشغيل فهي غير مستخدمة.
BATCH	عنـدمـا يبحث Plink86 plus عن اسم ملف أو اسم مكتبـة لربطه ولا يجده فإنه يتوقف مؤقتا ويظهر رسالة لادخال الاسم الصحيح للملف أو اسم الدليل الذي يحتويه بهذا الشكل: Can't find file xxxxxx.OBJ Enter new file name prefix (drive: or path name/) or . to quit =>. فإذا وضعت الاختيار BATCH داخل ملف LNK. فإن Plink86 plus يلغي عملية الـربط بدون أن يتوقف أو يظهر هذه الرسالة.
BEGINAREA	يمكن اختصار هذا الاختيار هكذا BEGIN ويستخدم عند استخدام مفهوم الاحلال (Overlay) قبل الأوامر التي تنشيء منطقة الاحلال (Overlay area) وهي أوامر SECTION. ويجب أن ينتـهـي هذا الاختيار بالاختيار ENDAREA. (راجـع الاحلال قبل نهاية هذا الفصل).

الاختيار	استخدامه
DEBUG	يعطي معلومات تفيد في أغراض تعقب وتصحيح أخطاء البرامج (Debugging). ويستخدم لاظهار رقم جزء البرنامج في حالة استخدام مفهوم الاحلال.
FILE	وتختصر هكذا FI يعطي لبرنامج الربط أسماء الملفات المطلوب ربطها. وإذا أردت استخدام أكثر من ملف OBJ. فيجب أن تفصل بين أسماء الملفات بعلامة , ولا تكتب الاسم الممتد OBJ. ضمن اسم الملف.
HEIGHT	يتحكم في عدد سطور الصفحة عند طباعة خريطة الذاكرة باستخدام Plink86 plus
LIBRARY	وتختصر هكذا LIB يعطي برنامج الربط أسماء المكتبات التي تشتمل على الأوامر والوظائف المستخدمة داخل البرامج والتي يجب أن تربط مع النظام أو البرنامج المطلوب ربطه. واسم الدليل الذي توجد تحته.
LOWERCASE	يطلب من برنامج الربط معاملة البيانات الموجودة في برنامج LNK. على أنها حروف صغيرة (Lower Case) حتى لو كان بينها حروف كبيرة.
MAP	يستخدم هذا الاختيار لطباعة خريطة لذاكرة الحاسب توضح نسبة استخدام الذاكرة وحجم ومكان البرامج الموجودة بها وأجزاء البرنامج في حالة استخدام overlay وهذا الاختيار يندر استخدامه إلا من قبل الذين يحتاجون لتصحيح برامج Assembly أو لاستخدام overlay وهذا الاختيار يتبعه أربعة اختيارت. ALL(A): يظهر أجزاء البرامج الموجودة بالذاكرة. Global(G): يظهر الرموز المستخدمة داخل البرنامج ومكانها داخل الذاكرة.

الاختيار	استخدامـــــــــــــه
	Modules(M): تظهر أسماء البرامج ومكانها في الذاكرة وحجمها بالاضافة إلى الرموز المستخدمة في البرنامج ومكانها داخل الذاكرة. Sections(S): يعطي معلومات مثل M لكنها أقل تفصيلا.
NOBELL	عندما يكتشف Plink86 plus خطأ فإنه يطلق صوت الجرس فإذا أردت ألا تسمع هذا الصوت عند حدوث خطأ استخدم هذا الاختيار.
OUTPUT	يعـطي برنـامـج Plink86 plus اسم الملف الذي سينتج بعد عملية الربط EXE. فإذا لم تستخدم هذا الاختيار فيخصص برنـامـج الربط اسم أول برنامج OBJ. في الملفات المطلوب ربطها للملف EXE. الذي ينتج.
OVERLAY	بعد ربط برامج OBJ. واستخراج ملف EXE. يفصل برنامج الربط المعلومات الموجودة في هذه البرامج إلى مجموعتين: البيانـات (Data) وهي البيانـات الثابتة مثل الرسائل وأسماء المتغيرات وأسماء الاختيارات. الأوامر (Code) وهي التعليمات التي يتضمنها البرنامج وعادة يضع Plink86 plus البيانات في الجزء الرئيسي من ملف EXE. فإذا أردت توفير مساحة الذاكرة من هذه البيانات استخدم أمر OVERLAY لهذا الغرض. (راجع بند الاحلال بعد قليل) فمثلا إذا أردت توجيه البيانات إلى الجزء الذي يوضع على القرص استخدم الاختيار هكذا: OVERLAY PROG, $CONSTANTS أما إذا أردت أن تترك البيانات في الجزء الرئيسي من البرنامج الذي يبقى دائما في ذاكرة الحاسب استخدم الاختيار هكذا: OVERLAY CODE

الاختيار	استخدامـــــــــــه
SECTION	يعرف اسم ملف OBJ. الذي سيستخدم في حالة الاحلال الداخلي.
SECTION INTO	يعرف اسم ملف OBJ. الذي سيستخدم في الاحلال الخارجي.
UPPERCASE	يعمل عكس الاختيار LOWERCASE
VERBOSE	يوجه Plink86 plus لاستخدام السطر رقم ٢٤ من الشاشة لاظهار معلومات عن عملية الربط.
WIDTH	عادة تظهر معلومات عن الذاكرة والبرامج الموجودة بداخلها بواسطة الاختيار MAP في حدود ٨٠ حرفا لكل عامود. فإذا أردت تغيير هذا العرض استخدم هذا الاختيار.
WORKFILE	يستخدم Plink86 plus ملفا خاليا ليسجل فيه معلومات بصفة مؤقتة حتى يتم الانتهاء من إنشاء ملف EXE. فإذا أردت توفير مساحة القرص خصوصا إذا كنت تستخدم قرصا مرنا فيمكنك توجيه Plink86 plus ليستخدم الذاكرة لهذا الغرض بدلا من ملف خال.

استخدام برنامج Plink86 plus من محث نظام التشغيل

من أشهر الطرق لاستدعاء Plink86 plus لربط برنامج أو برامج OBJ. استدعاء البرنامج مع الاختيارات المناسبة في سطر واحد تحت محث نظام التشغيل DOS. انظر المثال الآتي:

PLINK86 FI ADD, DEL, INQ LIB CLIPPER, EXTEND

وفي هذا المثال يتم استدعاء Plink86 بالاختيارات التالية:

١ ـ FI ADD, DEL, INQ

ومعناها أن ملفات الهدف (.OBJ) المطلوب ربطها هي ADD, DEL, INQ وهذه الملفات موجودة على الدليل الحالي لأننا لم نشر إلى دليل آخر.

٢ ـ LIB CLIPPER, EXTEND

ومعناها أن المكتبات التي سيحتاج إليها Plink86 هي CLIPPER و EXTEND وهذه المكتبات موجودة على الدليل الحالي لأننا لم نشر إلى دليل آخر.

وهذه أبسط طريقة لربط برنامج/برامج هدف (.OBJ) باستخدام Plink86 ما لم تكن في حاجة لاستخدام overlay.

استخدام Plink86 plus بالطريقة التبادلية (المباشرة)

يمكن استخدام الطريقة المباشرة (Interactive mode) لاستدعاء برنامج الربط بدلا من إدخال الاختيارات كلها في سطر واحد. وللتوضيح نسوق المثال الآتي:

بفرض أننا نريد ربط البرامج الآتية:

ADD DEL INQ

ولاستخدام هذه الطريقة اكتب Plink86 ثم اضغط مفتاح الادخال. سيتم تحميل برنامج Plink86 وسيظهر أمامك على الشاشة محث هكذا:

=>

أدخل الاختيارات التي ترغبها. كل اختيار في سطر مستقل ثم اضغط مفتاح الادخال بعد الانتهاء من كتابة كل سطر وعندما تنتهي من إدخال الاختيارات المطلوبة وترغب

في تنفيذ الربط اكتب علامة الفاصلة المنقوطة (semicolon) ثم اضغط مفتاح الادخال.

والمثال الآتي يستخدم هذه الطريقة لربط ثلاث برامج ووضعها في ملف جاهز للتنفيذ MENU.EXE

```
C:\CLIPPER>plink86
PLINK86plus ( Nantucket ) Version 2.24.
Copyright (C) 1987 by Phoenix Technologies Ltd.,
All Rights Reserved.
=>file add, del, inq
=>library clipper, extend
=>output menu
=>debug
=>verbose
=>;
```

وإليك شرح الاختيارات الواردة في هذا المثال

- FILE أو FI تخبر Plink86 أن أسماء ملفات الهدف OBJ. التي سيم ربطها هي ADD و DEL و INQ (وهي موجودة على الدليل الحالي).

- LIBRARY أو LIB تعلن أسماء المكتبات التي سيحتاج إليها برنامج الربط (وهي أيضا موجودة على الدليل الحالي).

- OUTPUT تطلب من Plink86 أن يضع الناتج في ملف جديد اسمه MENU.EXE

- DEBUG لاظهار معلومات عن البرامج الخطأ في حالة حدوث خطأ بأحدها.

- NOBELL لا داعي لسماع صوت الجرس في حالة حدوث خطأ أثناء عملية الربط.

- VERBOSE لاظهار معلومات عن عملية الربط والبرامج التي تستخدمها على الشاشة.

استخدام برنامج *plink plus* مع *LNK.*

الطريقة الثالثة لتشغيل Plink86 plus هي ملف نصي مكتوب بشفرة ASCII يخصص له الاسم الممتد LNK. بدلا من كتابتها في سطر واحد أو إدخالها بالطريقة التبادلية ولكي تستخدم ملف LNK. استخدم الصيغة التالية

```
PLINK86    @<linkfile>
```

حيث <linkfile> اسم ملف LNK. الذي يشتمل على الاختيارات اللازمة لعملية الربط.

انظر المثال الآتي وهو يشتمل على الاختيارات التي استخدمناها في المثال السابق بالاضافة إلى السطر الأول وهو عبارة عن تعليق أو ملاحظة فقط

```
# This is a sample link file
fi     add, del, inq
lib    clipper, extend
output menu
debug
nobell
verbose
```

فبفرض أن هذه الاختيارات موجودة في ملف اسمه Menu.LNK فإذا أردت تنفيذ الاختيارات أدخل الأمر الآتي:

```
PLINK86    @Menu
```

وتستطيع كتابة ملف Menu.LNK بأي منسق للكلمات أو حتى بمحرر السطر الموجود في DOS

استخدام ملف تجميعي لترجمة وربط الملفات

في التطبيقات المعقدة والكبيرة يحسن تجميع أوامر ترجمة وربط الملفات ووضعها في ملف تجميعي بحيث يتم تنفيذها بمجرد استدعاء الملف للتنفيذ من محث DOS.

وهـذه الطريقة هي التي استخدمها مصممو «كلبر» لترجمة وربط الملفات التي تبدأ بحروف dbu في ملف MAKEDBU.BAT ولترجمة وربط الملفات التي تبدأ بحرفي rl في ملف MAKERL.BAT.

من محث نظام التشغيل تحت دليل «كلبر» أدخل الأمر التالي:

TYPE MAKEDBU.BAT

واضغط مفتـاح الادخـال بعـد الانتهاء من كتابته. ستحصل على محتويات الملف مكتوبة بشفرة ASCII. انظر شكل ١-٦ ومنه تلاحظ أن السطور الثمانية الأولى لترجمة ٨ ملفات كل ملف بصفة مستقلة مع الاختيارات التالية:

```
clipper dbu -m -q -l
clipper dbuview -m -q -l
clipper dbustru -m -q -l
clipper dbuedit -m -q -l
clipper dbuindx -m -q -l
clipper dbucopy -m -q -l
clipper dbuutil -m -q -l
clipper dbuhelp -m -q -l
plink86 fi dbu,dbuview,dbustru,dbuedit,dbuindx,dbucopy,dbuutil,dbuhelp
                         lib \clipper\clipper,\clipper\extend
```

شكل ١-٦

m-: وتعني ترجمة هذا البرنامج فقط.
q-: ومعناه عدم ظهور رقم السطر أثناء عملية الترجمة.
l-: ومعناه ترجمة البرنامج بدون إضافة رقم السطر إلى برنامج OBJ. والسطر الأخير خاص بربط الملفات الثمانية واستخراج ملف جاهز للتنفيذ EXE. باسم dbu.EXE (الاسم الأول في المجموعة).

اطبع ملف MAKERL.BAT بنفس الطريقة ستحصل على شكل ٢-٦ وهو أيضا خاص بترجمة الملفات الثلاثة التي تبدأ بحرفي rl ثم ربطها معها واستخراج ملف جاهز EXE. باسم rl.exe

```
cclipper rlfront -q -m
clipper rlback -q -m
clipper rldialog -q -m
plink86 fi rlfront,rldialog,rlback lib \clipper\clipper,\clipper\extend
                                output rl
```

شكل ٢ـ٦

فائدة: إذا نفذت كلا من البرنامجين السابقين: makedbu.bat و makerl.bat ستحصل على برنامجي dbu.exe و rl.exe وسنشرح كلا منهما فيما بعد.

إلا أننا يمكننا كتابة ملف تجميعي بحيث يشتمل على معطيات يعوض عنها من خارجه لتحقيق مرونة أكثر.

انظر المثال التالي:

المثال عبارة عن ملف تجميعي اسمه ANYFILE.BAT يشتمل على سطرين هما:

```
CLIPPER %1
IF NOT ERRORLEVEL1 PLINK86 FI %1 CLIPPER, EXTEND
```

* **السطر الأول** علامة 1% معناها أن يقبل البرنامج قيمة من خارجه أثناء التنفيذ. وبالتالي سيقوم بترجمة البرنامج الذي يكتب اسمه بعد اسم ملف ANYFILE فيما بعد.

* **والسطر الثاني** معناه إذا لم يحدث خطأ أثناء عملية الترجمة فيتم استدعاء برنامج Plink86 لربط الملف الذي تمت ترجمته مع تخصيص مكتبة CLIPPER.LIB ومكتبة EXTEND.LIB ليبحث فيهما PLINK86 عن التعليمات والوظائف المطلوبة.

فإذا أردت ترجمة وربط برنامج باسم MENU.PRG مع كل البرامج التي ينادى عليها والمرتبطة به والحصول على ملف جاهز للتنفيذ باسم MENU.EXE، أدخل

الأمـر الآتي من محث DOS (بشرط أن يكون ملف ANYFILE.BAT موجودا على الدليل الحالي).

ANYFILE MENU

وإذا أردت في مرة قادمة ترجمة وربط ملف آخر وليكن اسمه MASTER.PRG أدخل الأمر بهذه الصورة

ANYFIL MASTER

استخدام الاحلال
Using Overlay

أحيانا نحتاج لتشغيل برامج كبيرة تزيد المساحة المطلوبة لتشغيلها عن المساحة المتاحة في ذاكرة الحاسب. وفي هذه الحالة لن نستطيع تشغيل هذا البرنامج إلا بواحدة من اثنين:

الطريقة الأولى: زيادة حجم الذاكرة لتناسب حجم البرنامج وهذه طريقة غير عملية لأنك لا تملك أن تطلب من مستخدمي برامجك التي تباع وتسوّق تغيير حاسباتهم لتلائم برامجك. بل الأفضل أن تلائم أنت برامجك لتتمشى مع حاسباتهم بإمكانياتهم.

الطريقة الثانية: وهي الطريقة العملية التي تلائم البرامج الكبيرة لتتمشى مع حجم الذاكرات الصغيرة وتسمى هذه الطريقة في علم الكمبيوتر overlay programming أي البرمجة بطريقة الاحلال. ومن المزايا الهامة في «كلبر» أنها تسمح لك باستخدام طريقة الاحلال (overlay).

وقبل شرح الاحلال باستخدام «كلبر» نوضح فيما يلي عدة تعريفات لحاجتنا إليها عند شرح هذا الموضوع.

ملاحظة: الصفحات التالية من هذا الفصل تخاطب ذوي الخبرة الطويلة بالبرمجة والذين يطورون أنظمة لغرض بيعها في الأسواق. فإذا كنت مبتدئا في البرمجة ننصحك بتخطي هذه الصفحات والعودة إليها عندما تحتاج لذلك.

البرمجة بطريقة الاحلال (Overlay programming)

هي أسلوب من أساليب كتابة البرامج الكبيرة التي تزيد مساحتها عن المساحة المتاحة في الذاكرة حيث يتم تقسيم البرنامج إلى سلسلة أجزاء شبه متكاملة وعند التنفيذ يتم تنفيذ كل جزء على حده بحيث يتم إحلال الجزء المراد تنفيذه في نفس المكان بالذاكرة الذي كان يشغله الجزء السابق. تستخدم كثير من البرامج الكبيرة مثل dBASE III PLUS أو dBASE IV أسلوب البرمجة بطريقة الاحلال.

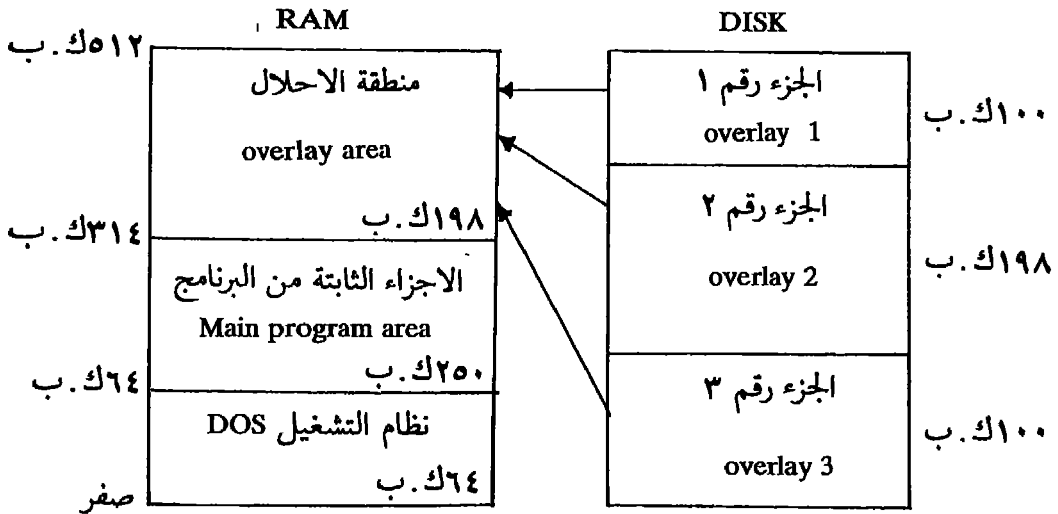
منطقة الاحلال Overlay area

هي تلك المساحة من الذاكرة الرئيسية لجهاز الحاسب التي يتم تخصيصها لاحلال أجزاء البرامج محل بعضها عند التنفيذ.

فكرة الاحلال

يقصد بالاحلال (overlay) تشغيل برنامج ذي حجم أكبر من الحجم المتاح له على الذاكرة الداخلية للحاسب RAM وذلك بتقسيم البرنامج إلى أجزاء وتخزين الجزء من البرنامج الذي لم تتسع له الذاكرة الداخلية على وسيط خارجي غالبا ما يكون وحدة القرص الصلب (Hard diks) وعند الحاجة لذلك الجزء يتم نقله من القرص الصلب إلى الذاكرة الداخلية للحاسب. فمثلا إذا كانت المساحة التي يشغلها برنامج ما هي ٧٢٢ ك.ب. وكانت مساحة الذاكرة المتاحة هي ٥١٢ ك.ب. فيتحتم استخدام البرمجة بطريقة الاحلال لتشغيل هذا البرنامج على جهاز الحاسب.

ويوضح شكل ٣-٦ رسما تخطيطيا يوضح طريقة عمل البرامج التي تزيد مساحتها عن مساحة ذاكرة الحاسب وذلك بتقسيمها إلى أجزاء تتبادل الذاكرة واحدا تلو الآخر.



شكل ٣-٦

وفي هذا الشكل فإن كل أجزاء البرنامج الموجودة على القرص + الجزء الرئيسي الذي يجب أن يبقى في الذاكرة طول الوقت تزيد عن المساحة المتاحة في ذاكرة الحاسب.

ويحدد الجزء الثاني من أجزاء البرنامج مساحة منطقة الاحلال لأنه هو أكبر جزء سيوضع في هذه المنطقة أما الجزء الأول والثالث فهما أصغر بكثير من مساحة منطقة الاحلال.

ولتوضيح البرمجة بطريقة الاحلال نقول إن أجزاء البرنامج التي لا يتعامل معها النظام تبقى محفوظة على القرص الممغنط حتى يتم استدعاؤها من داخل البرنامج فيحل الجزء المطلوب مكان الجزء الموجود بمنطقة الاحلال. ويعود الجزء الذي كان موجودا في منطقة الاحلال إلى القرص الممغنط وبهذا نوفر مساحة الذاكرة المطلوبة لتشغيل البرنامج كله.

إلا أننا ننصح بعدم استخدام هذا الأسلوب إلا في حالة الضرورة القصوى لأن نقل أجزاء البرامج من القرص الصلب إلى ذاكرة الحاسب يستغرق وقتا طويلا نسبيا مما يتسبب في بطء تنفيذ هذه البرامج.

إنشاء منطقة الاحلال

سنتكلم عن إنشاء نوعين من الاحلال:

- الاحلال الداخلي Internal overlay
- الاحلال الخارجي External overlay

الاحلال الداخلي

باستخدام هذه الطريقة يتم تقسيم ملف EXE. نفسه إلى أجزاء وتوضع الأجزاء الغير مطلوبة للتنفيذ على القرص الممغنط حتى يتم استدعاؤها من داخل البرنامج فيتم تحميلها داخل الذاكرة في منطقة الاحلال. وميزة هذه الطريقة أن النظام الذي ترغب في بيعه أو تسويقه يبقى موجودا داخل ملف واحد (EXE.) إلا أن الملف يقسم إلى أجزاء ساعة التنفيذ. ولابد أن تكون مساحة القرص الذي يوضع عليه الملف أكبر من مساحة الذاكرة.

والمثال الموجود في شكل ٤-٦ يشتمل على ملف من نوع LNK. اللازم لتقسيم برنامج INVNTRY إلى أجزاء وربطه مع نظام التشغيل.

```
# Link file for creating overlay
FI INVNTRY
LIB CLIPPER,EXTEND
OVERLAY PROG,$CONSTANTS
BEGINAREA
  SECTION FILE PROG1,PROG2,PROG3
  SECTOPN FILE INQ,REP,LBL
  SECTION FILE UTY,BAK
ENDAREA
```

شكل ٤-٦

ونوضح فيما يلي الأوامر التي وردت في هذا الملف والتي توجه plink 86 plus لانشاء منطقة الاحلال.

- السطر رقم ١ لتوضيح غرض الملف وهو لا يخاطب Plink86 plus
- السطر رقم ٢ يشتمل على اسم الملف المطلوب ربطه باستخدام Plink86 plus
- الأمر LIB في السطر رقم ٣ يوضح لبرنامج Plink86 plus أسماء المكتبات التي ستلزمه.
- الأمر OVERLAY في السطر رقم ٤ يطلب وضع البيانات الثابتة مثل الرسائل وأسماء حقول الذاكرة الواردة بالبرنامج في منطقة الاحلال بعيدا عن الذاكرة الرئيسية حتى يتم استدعاؤها.
- الأمر BEGINAREA في السطر رقم ٥ خاص بإنشاء منطقة الاحلال.
- الأمر SECTION الموجود في السطر رقم ٦ يشتمل على أسماء برامج الهدف (object files) التي سيتم ربطها معا لتكون في منطقة إحلال واحدة وهي PRG1 - PRG2 - PRG3
- الأمر SECTION الموجود في السطر رقم ٧ يطلب وضع البرامج LBL - REP - INQ في منطقة إحلال واحدة.

- الأمر SECTION الموجود في السطر رقم ٨ يطلب ربط برنامجي BAK UTY ووضعها داخل منطقة واحدة على القرص.
- الأمر ENDAREA في السطر رقم ٩ يخبر Plink86 بنهاية التقسيم المطلوب.

الاحلال الخارجي

يعامل الاحلال الخارجي معاملة الاحلال الداخلي والفرق بينهما أن الاحلال الخارجي يستخدم عندما يكون النظام أو البرنامج المطلوب ربطه باستخدام Plink86 plus أكبر من مساحة القرص الممغنط ولذلك يوضع على أكثر من قرص واحد. وبالتالي فإن plink 86 plus يبحث عن أكثر من ملف بدلا من أن يبحث في ملف واحد. ويكتب الاحلال الخارجي على ملف يخصص له الاسم الممتد OVL.

والمثال الموجود في شكل ٥ـ٦ يشتمل على ملف LNK. الذي يقوم بإنشاء إحلال خارجي. ومنه تلاحظ أن هذا الملف يشبه الملف اللازم لانشاء إحلال داخلي باستثناء فرقين اثنين:

```
# Link file for creating external overlay
FI INVNTRY
LIB CLIPPER,EXTEND
OVERLAY PROG,$CONSTANTS
BEGINAREA
  SECTION INTO OVLY1.OVL FILE PROG1,PROG2,PROG3
  SECTOPN INTO OVLY2.OVL FILE INQ,REP,LBL
  SECTION INTO OVLY3.OVL FILE UTY,BAK
ENDAREA
```

شكل ٥ـ٦

الأول: استخدمنا أمر SECTION INTO بدلا من أمر SECTION لانشاء ملف إحلال (overlay file) واشتمل الأمر على اسم ملف الاحلال بالاضافة إلى البرامج التي سيتم ربطها معا لتوضع داخل هذا الملف. فمثلا الأمر SECTION INTO الموجود في السطر الخامس ينشيء ملف إحلال اسمه OVLY1.OVL ليشتمل على

البرامج PRG1.OBJ و PRG2.OBJ و PRG3.OBJ

الثاني: سيتم إنشاء ثلاثة ملفات جديدة بالاضافة إلى ملف INVNTRY بالأسماء الآتية:

OVLY1.OVL

OVLY2.OVL

OVLY3.OVL

وسيقل حجم ملف INVNTRY بمقدار الأوامر التي ستنتقل إلى الملفات الجديدة.

تنظيم تقسيم البرامج *Overlay Management*

عادة يتم كتابة البرامج التي ستستخدم نظام الاحلال بنفس الطريقة التي تكتب بها البرامج الصغيرة الأخرى فإذا اكتشفت أن مساحة البرنامج أكبر من مساحة الذاكرة فعليك استخدام إحدى الطريقتين اللتين شرحناهما فيما سبق حسب نوع الاحلال الذي يناسبك وفيما يلي نوجهك إلى عدة نصائح لتجنب الأخطاء التي قد تقع فيها.

- يجب أن تتأكد أن أجزاء البرنامج (overlays) سواء كان برنامجا واحدا مجزءا إلى أجزاء (وهو الاحلال الداخلي) أو برنامجا مقسما إلى عدة ملفات (وهو الاحلال الخارجي) لا ينادي أحدها على الآخر. لأن منطقة الاحلال لن تتسع لكلا الجزئين في وقت واحد. وتحدد مساحة منطقة الاحلال تبعا لمساحة أكبر جزء أو ملف سيوضع فيها (راجع شكل ٣-٦).
- إذا كان هناك برامج أو وظائف خاصة يستخدمها أكثر من (overlay) يجب وضعها ضمن الجزء الرئيسي من البرنامج الذي يبقى ثابتا بالذاكرة طول الوقت.
- يجب ألا تلجأ إلى برمجة الاحلال إلا في حالات الضرورة القصوى تجنبا لبطء التنفيذ الذي تسببه.
- لكي تتأكد أن أحد أجزاء البرنامج يعمل بصفة مستقلة ولا يشترك مع جزء آخر في أمر أو أوامر تسبب استدعاء الجزئين معا إلى منطقة الاحلال جرب ربطه في ملف EXE. منفردا وتجربته مستقلا.

ـ إذا أمكن ضم جزئين أو أكثر من البرنامج المطلوب تجزئته إلى بعضهما ووضعهما داخل منطقة الاحلال فيجب وضعهما داخل جزء واحد (one overlay) لأنه كلما نقص وقت تحميل الأجزاء إلى الذاكرة وإرجاعها إلى القرص كلما زادت سرعة تنفيذ النظام أو البرنامج.

ـ إذا استخدمت الاحلال الخارجي (External overlay) الذي ينشيء أكثر من ملف فيجب التأكد من أن المساحة المخصصة لكل ملف مناسبة لمساحة منطقة الاحلال (overlay area). وذلك لأن الملفات التي تأخذ مساحة أكبر من مساحة منطقة الاحلال لن يمكن تشغيلها. أما الملفات الصغيرة فيفضل ضمها مع ملفات أخرى لتحقيق السرعة المطلوبة أثناء تنفيذ النظام وللوصول إلى ذلك هناك طريقتان:

الطريقة الأولى: تتلخص في ربط الملفات مع بعضها ثم ملاحظة حجمها تحت محث DOS باستخدام DIR (انظر شكل ٦ـ٦).

```
Volume in drive C is DISK1_VOL1

Directort of C:\DBMS
XXX          OVL        43986      22/11/90      12:00
YYY          OVL        59936      22/11/90      12:00
ZZZ          OVL        63280      22/11/90      12:00
          3 Files(s)   1729088 bytes free
```

شكل ٦ـ٦

وهي التي تناسب مبرمجي «دي بيس» ومن ليست لهم خبرة بلغات البرمجة الدنيا مثل Assembly

الطريقة الثانية: إنشاء خريطة للذاكرة توضح أسماء الملفات ومكان كل منها داخل الذاكرة والحجم المخصص لكل منها داخل الذاكرة. . . الخ.

ويستخـدم لذلك أمر MAP مع Plink86 plus ولأن النتائج التي نحصل عليها تظهر دائما بالنظام السداسي عشر (Hexadecimal). لذلك فهذه الطريقة لا تناسب إلا من لهم خبرة طويلة بالبرمجة باللغات الدنيا وتحويل الأرقام من النظام السداسي عشر إلى النظام العشري.

تــذكــر...!

يتم تحويل البرامج المصدرية (PRG.) إلى صورة جاهزة للتنفيذ (EXE.) باتباع خطوتين:

- الأولى: ترجمة البرامج المصدرية (Source Code) إلى برامج هدف (Object Code).
- الثانية: ربط برامج الهدف التي نحصل عليها من الخطوة الأولى مع بعضها ومع الوظائف التي تتطلبها من مكتبة البرامج باستخدام برنامج الربط Plink86 Plus.

وقد شرحنا الاختيارات السبعة التي تتحكم في برنامج الترجمة واستخدام ملف CLP. وشرحنا الطرق المختلفة لاستخدام برنامج الربط Plink86 Plus سواء من محث نظام التشغيل أو من خلال ملف LNK. بالاضافة إلى الاختيارات التي تتحكم في برنامج الربط Plink86 Plus.

وأخيرا شرحنا كيفية استخدام ملف تجميعي (Batch file) لترجمة وربط أكثر من ملف. وإتماما للفائدة أوضحنا كيفية تقسيم البرامج الكبيرة إلى أجزاء صغيرة أثناء التنفيذ أو تقسيم النظام الكبير إلى أكثر من ملف باستخدام مفهوم الاحلال (Overlay).

الباب الثاني

مفاهيم متقدمة

الفصل السابع

المصفوفات Arrays

يشرح هذا الفصل معنى المصفوفات وضرورة استخدامها ثم يشرح الأوامر والوظائف التي تتعامل مع المصفوفات مثل إنشاءها، وتعبئة عناصرها بقيم مختارة أو بمعلومات عن الملفات أو بمواصفات حقول الملف، ونسخها، وحذف أحد عناصرها وترتيب عناصرها، والبحث فيها ثم خصصنا الوظيفة ACHOICE() بشرح منفرد نظرا لأهميتها.

بالاضافة إلى تشغيل البيانات الموجودة في شكل سجلات داخل ملفات مخزنة على قرص ثابت أو مرن يمكن تشغيل بيانات مخزنة داخل جداول أو مصفوفات يتم وضعها بالذاكرة أثناء التشغيل. إذن الجداول أو المصفوفات عبارة عن مجموعة من البيانات تخزن تحت مظلة واحدة داخل ذاكرة الحاسب. ويطلق على المصفوفات أو الجداول كلمة Arrays.

ملاحظة: أحيانا تترجم Array جدول وأحيانا تترجم مصفوفة إلا أننا اخترنا ترجمتها إلى مصفوفة لأن «كلبر» تتعامل مع المصفوفة ذات البعد الواحد والتي تختلف عن الجدول الذي يتكون من بيانات في شكل سطور وأعمدة.

ويمكن تقسيم المصفوفات عموما إلى نوعين:

- مصفوفة ذات بعد واحد مثل الأنواع التي تتعامل معها «كلبر» وفيها تخزن البيانات في شكل سطر داخل الذاكرة.
- مصفوفة ذات بعدين مثل الأنواع التي تتعامل معها «دي بيس فور» أو «بيسك» أو «كوبول» أو غيرها وفيها تخزن البيانات في شكل جدول ذي أعمدة وسطور.

ولا يستغني إعداد نظم إدارة قواعد البيانات عن استخدام المصفوفات والتعامل معها. فمثلا في برامج إعداد مرتبات الموظفين قد نحتاج لخصم الضرائب أو التأمينات المستحقة من مرتبات الموظفين بنسب مختلفة طبقا لشرائح مرتباتهم. ولأن عدد شرائح المرتبات غالبا ما تكون قليلة فيفضل تخزين معدلات الضرائب أو التأمينات الخاصة بكل شريحة داخل مصفوفة بالذاكرة بدلا من تخصيص ملف قاعدة بيانات مستقل لها كما كنا نفعل في قاعدة البيانات dBASE III PLUS ويتم البحث داخل المصفوفة المخزنة بذاكرة الحاسب عن المعلومة المطلوبة وهي معدل الضريبة في هذا المثال.

وتسمح قاعدة البيانات Clipper بإنشاء مصفوفات يصل عددها إلى ٢٠٤٨ مصفوفة وتتكون كل مصفوفة من مجموعة عناصر (Elements) لا يزيد عددها عن ٤٠٩٦ عنصرا. ويتم التعامل مع كل عنصر موجود داخل المصفوفة كما لو كان حقل ذاكرة مستقل. ولذلك فيمكن أن تشتمل المصفوفة الواحدة على أكثر من نوع بمعنى

أن عناصرها يمكن أن يكون بعضها حرفيا وبعضها رقميا وبعضها تاريخيا وبعضها منطقيا.

ولتوضيح أهمية استخدام المصفوفات نفترض أن مؤسسة تتعامل مع ٥ محصلين ويتعامل كل محصل مع ٥٠٠ عميل ونريد أن نعرف من حين لآخر عملاء محصل معين. في هذه الحالة باستخدام «دي بيس ثري بلاس» وبدون استخدام المصفوفات يتم تسجيل أسماء المحصلين على ملف قاعدة بيانات مستقل أو يتم إضافة أسمائهم إلى ملف العملاء الذي يشتمل على آلاف السجلات. وبالتالي سيتكرر حقل اسم المحصل في كل السجلات بدون داع مما يتسبب في زيادة المساحة المخصصة للملف على القرص. أما باستخدام المصفوفات التي تستخدمها «كلبر» فيمكننا تخزين أسماء المحصلين الخمسة داخل مصفوفة واحدة يتم إنشاؤها بالذاكرة ويقال عن كل محصل عنصر داخل المصفوفة أو Element. فبفرض أن اسم المحصل الأول محمد واسم المحصل الثاني أحمد والمحصل الثالث عبدالله والمحصل الرابع ميسرة والمحصل الخامس عمرو فإذا أردت أن تظهر اسم عبدالله فيكفي أن تشير إليه برقمه داخل المصفوفة وهو رقم ٣.

بهذا نستطيع القول إن عناصر المصفوفة تشبه حقول السجل إلا أنها تخزن داخل ذاكرة الحاسب أما حقول السجل فتخزن داخل ملف على قرص ثابت أو مرن.

ويجب أن تحدد عدد عناصر المصفوفة عند إنشائها وقبل تعبئة بياناتها. وبالرغم من أن «كلبر» تنشىء مصفوفة ذات بعد واحد إلا أنه يمكننا بقليل من التحايل التعامل مع مصفوفة في شكل جدول ذي أعمدة وسطور. وذلك بإنشاء أكثر من مصفوفة لهذا الغرض فمثلا إنشاء ٣ مصفوفات تشتمل كل منها على ١٢ عنصرا ينتج في النهاية مصفوفة أو جدول يشتمل على ٣ سطور و١٢ عامودا.

وسوف نتناول فيما يلي الأوامر والوظائف التي تتعامل مع المصفوفات مثل إنشائها وتعبئة عناصرها ونسخها وحذف أحد عناصرها وترتيب عناصرها والبحث فيها وسنخص بالشرح الوظيفة ()ACHOICE نظرا لأهميتها بالاضافة إلى أمثلة وافية توضح كيفية استخدام هذه الوظائف والأوامر داخل البرامج.

إنشاء المصفوفة

تستخدم «كلبر» أمر DECLARE لانشاء المصفوفة. فإذا أردت أن تنشىء مصفوفة باسم Arr1 تتكون من ٥ عناصر استخدم هذا الأمر

DECLARE ARR1[5]

ولابد من استخدام هذه الأقواس [ ] لتفهم كلبر أن ما بداخلها هو عدد عناصر المصفوفة.

ويمكن إنشاء أكثر من مصفوفة داخل الأمر الواحد. انظر المثال الآتي:

DECLARE ARR1[5], ARR2[10], ARR3[13]

ويمكن تخزين اسم المصفوفة وعدد عناصرها داخل حقل ذاكرة والتعامل مع محتويات الذاكرة مباشرة. انظر المثال التالي:

```
ARVAR = "ARR1"
ELVAR = 5
DECLARE &ARVAR[ELVAR]
```

وهذا المثال يساوي استخدام الأمر هكذا:

```
DECLARE ARR1[5]
```

إلا أنه لا يمكننا التعامل مع الأقواس الموجودة داخل حقل الذاكرة فمثلا لا يمكننا استخدام المثال الآتي بديلا عن المثال السابق.

```
ARFULL = "ARR1[5]"
DECLARE &ARFULL
```

ويتحدد طول المصفوفة بعدد عناصرها فمثلا الأمر التالي:

```
? LEN(ARR1)
```

يعطي نتيجة 5

وتعامل المصفوفات معاملة حقول الذاكرة التي تنشأ PRIVATE ما لم تنص على أنها PUBLIC.

ملاحظة: راجع أمر PUBLIC وأمر PRIVATE في الباب الثالث الذي يشرح أوامر «كلبر».

المثال التالي ينشيء مصفوفة باسم DAYSتشتمل على ٧ عناصر وينشئها PUBLIC

```
PUBLIC DAYS[7]
```

أما أمر PRIVATE DAYS[7] فهو مساو للأمر

```
DECLARE DAYS[7]
```

لأن المصفوفات تنشأ تلقائيا PRIVATE.

تعبئة عناصر المصفوفة

تعد المصفوفة كلها من وجهة نظر «كلبر» بحقل ذاكرة واحد أما عناصر المصفوفة فيمكن أن تكون خليطا من بيانات حرفية أو رقمية أو تاريخية أو منطقية. ولكي تضع قيمة داخل عناصر المصفوفة استخدم أمر STOREأو علامة =مثلما تفعل لكي تضع قيمة داخل حقل ذاكرة.

المثال التالي يضع بيانات من أنواع مختلفة داخل المصفوفة ARR1التي أنشأناها في المثال السابق.

```
ARR1[1] = "Test array"
ARR1[2] = 64
ARR1[3] = .F.
ARR1[4] = DATE()
ARR1[5] = "Nothing"
```

ومن هذا المثال يتضح لنا أن التعامل مع أحد عناصر المصفوفة يتم بتحديد رقمه فمثلا الأمر ARR1[3]=.F.معناه تخزين القيمة المنطقية .F.داخل العنصر الثالث من المصفوفة المسماة ARR1وبالمثل الأمر ARR1[3] ? سيظهر القيمة .F.أما الأمر TYPE("ARR1[3]") ?ومعناه ما هو نوع البيانات المخزنة داخل العنصر الثالث

من المصفوفة ARR1 فيعطي النتيجة L بمعنى Logical ويمكن استدعاء إجراء أو وظيفة خاصة للتنفيذ باستخدام محتويات أحد عناصر المصفوفة أو حتى المصفوفة كلها كمعـطيات للاجراء أو الوظيفة. فمثلا إذا أردت أن تستدعي الاجراء PROCA للتنفيذ مستخدما تاريخ اليوم استخدم هذا الأمر

```
DO  PROCA WITH ARR1[4]
```

كما يمكن استخدام الأمر بالصيغة التالية إذا أردت استخدام المصفوفة كلها

```
DO  PROCA WITH ARR1
```

استخدام الوظيفة ()AFILL لتعبئة عناصر المصفوفة

إذا أردت أن تضع قيمة واحدة داخل كل عناصر المصفوفة استخدم الوظيفة ()AFILL لهذا الغرض لأنها أسرع طريقة لتعبئة كل عناصر المصفوفة بقيمة واحدة. ويمكن استخدام هذه الوظيفة لتعبئة كل عناصر المصفوفة أو لتعبئة بعض عناصرها.

المثال التالي يضع القيمة صفر داخل كل عناصر المصفوفة ARR2

```
DECLARE ARR1[5],ARR2[20],ARR3[13]
AFILL(ARR1,0)
```

ويمكن إضافة اختيارين آخرين للوظيفة ()AFILL الأول يحدد رقم أول عنصر يتم تعبئته والثاني يحدد عدد العناصر التي سيتم تعبئتها.

المثال التالي يضع القيمة صفر داخل العناصر من ٥ إلى ١٥ داخل المصفوفة ARR2

```
AFILL(ARR2,0,5,10)
```

ويمكن باستخدام أمر FOR...NEXT بديلا للوظيفة ()AFILL لتعبئة عناصر المصفوفة. انظر المثال التالي وهو مساو للأمر AFILL (ARR1,0)

```
FOR ELM_NO = 1 TO 5
  ARR1[ELM_NO] = 0
NEXT
```

نسخ المصفوفة

استخـدم الـوظيفـة ()ACOPY لنسخ مصفوفة إلى مصفوفة أخرى. وهذه الوظيفة تسمح بنسخ كل أو بعض عناصر المصفوفة إلى مصفوفة أخرى.

المثال التالي ينسخ مصفوفة باسم OLD إلى أخرى باسم NEW

```
LENGTH = 10
DECLARE OLD[LENGTH],NEW[LENGTH]
ACOPY(OLD,NEW)
```

أما إذا كنت قديرا بالبرمجة وتريد أن تظهر مهارتك الشخصية لنسخ مصفوفة إلى أخرى بدون وظيفة()ACOPY استخدم المثال التالي بدلا من()ACOPY في المثال السابق.

```
FOR ELM_NO = 1 TO LENGTH
   NEW[ELM_NO] = OLD[ELM_NO]
NEXT
```

ويمكن نسخ بعض عناصر المصفوفة وذلك بإضافة ٣ اختيارات أخرى إلى الوظيفة ()ACOPY وهي:

١ ـ مكان بداية النقل في المصفوفة القديمة.

٢ ـ عدد العناصر التي سيتم نقلها.

٣ ـ مكان بدايةا لنقل في المصفوفة الجديدة.

فإذا أردنا نسخ العناصر الثلاثة الأولى من المصفوفة OLD إلى العناصر الثلاثة الأخيرة في المصفوفة NEW استخدم المثال التالي:

```
AFILL(OLD,0)
AFILL(NEW,9)
ACOPY (OLD,NEW,1,3,8)
```

في هذا المثال وضعنا الرقم صفر في كل عناصر المصفوفة OLD والرقم ٩ في كل عناصر المصفوفة NEW ثم استخدمنا()ACOPY للنسخ.

وبعد تنفيذ الأمر فإن العناصر ٨، ٩، ١٠ ستشتمل على الرقم صفر.

حذف أحد عناصر المصفوفة

تستخـدم الـوظيفـة ADEL() لحذف أحـد عناصر المصفوفة. وبعد عملية الحذف يحل العنصر التالي للعنصر المحذوف محله ويصير الأخير غير معروف لكلبر لأنه لم يعد يشتمل على شيء. انظر المثال الآتي:

```
DECLARE NAMES[4]
NAMES[3] = "Ali"
ADEL(NAMES,2)
```

في هذا المثـال تم حـذف العنصر الثاني من المصفوفة NAMES وبالتالي فإن العنصر الثالث حل محله والرابع أصبح غير معروف ولذلك فإن أمر:

```
? NAMES[2]
```

سيظهر اسم Ali

أما أمر

```
? NAMES[4]
```

فسيعطيك رسالة خطأ لأن العنصر الرابع لم يعد موجودا.

إدخال عنصر بين عناصر المصفوفة

تستخدم الوظيفة AINS() لحشر عنصر بين عناصر موجودة داخل مصفوفة. والعنصر الجديد لا يشتمل على أية بيانات. ويتسبب في إزاحة باقي عناصر المصفوفة ولذلك يجب أن تكون المصفوفة كافية لاحتوائه وإلا فإن العنصر الأخير سيضيع ويجب تعبئـة العنصر الجديد قبل استخدامه لأن استخدام عنصر لا يشتمل على بيانات يسبب خطأ.

المثال التالي يدخل عنصرا جديدا مكان العنصر الثاني وبالتالي يزاح العنصر الثاني والثالث مكان الثالث والرابع.

```
DECLARE EXPAND[4]
EXPAND[1] = "Magdi"
EXPAND[2] = "Ali"
EXPAND[3] = "Abdu"
AINS(EXPAND,2)
? EXPAND[3]
```

لاحظ أن السؤال عن محتويات العنصر الثالث بعد إضافة العنصر الجديد سيظهر لنا محتويات العنصر الثاني قبل الاضافة ولذلك سنحصل على اسم Ali في هذا المثال.

ترتيب عناصر المصفوفة

تستخدم الوظيفة ()ASORT لترتيب عناصر المصفوفة ترتيبا تصاعديا.

المثال التالي يرتب عناصر المصفوفة طبقا لأبجديات الأسماء:

```
DECLARE NAMES[4]
NAMES[1] = "Magdi"
NAMES[2] = "Ali"
NAMES[3] = "Yaser"
NAMES[4] = "Naser"
ASORT(NAMES)
```

فإذا أردت أن تسأل عن محتويات العنصر الأول بعد الترتيب ستحصل على اسم Ali والعنصر الأخير سيشتمل على اسم Yaser

```
? NAMES[1]          && Ali اجابة كلبر:
? NAMES[4]          && YASER اجابة كلبر:
```

ويمكن ترتيب بعض عناصر المصفوفة إذا أضفت إلى الوظيفة اختيارين هما رقم العنصر الذي سيبدأ عنده الترتيب وعدد العناصر التي سيتم ترتيبها.

المثال التالي يرتب عناصر المصفوفة SORTED ابتداء من العنصر رقم ١٠ إلى العنصر رقم ٢٥.

```
ASORT(SORTED(10,16)
```

البحث داخل المصفوفة

تبحث الوظيفة ()ASCAN عن أول عنصر داخل مصفوفة يتطابق مع عبارة معينة.

المثال التالي يبحث في مصفوفة تتكون من ٢٠ عنصرا تشتمل على أرصدة الموردين الدائنة عن الرصيد 150000

```
DECLARE VENDORS[20]
? ASCAN(VENDORS,150000)
```

فبفرض أن ترتيب الرصيد هو العاشر داخل المصوفة فستحصل على الرقم ١٠.

ويمكن البحث داخل بعض عناصر المصفوفة فقط إذا حددنا رقم العنصر الذي يبدأ البحث عنده وعدد العناصر التي سيتم بحثها.

المثال التالي يبحث داخل العناصر من ١٠ إلى ١٥ فقط داخل المصفوفة السابقة.

```
? ASCAN(VENDORS,150000,10,6)
```

تعبئة المصفوفة ببيانات عن الملفات

تستخدم الوظيفة ()ADIR لغرضين هما: معرفة عدد الملفات الموجودة على الدليل الحالي والتي تتطابق مع الرموز المعطاة، ونقل معلومات عن هذه الملفات مثل حجمها وتاريخ ووقت إنشائها إلى مصفوفات أخرى. وهذه المعلومات هي التي تظهر عند إصدار أمر DIR تحت تحت نظام التشغيل DOS. ويستخدم الرمزان الشاملان *

أو ? للدلالة على أسماء الملفات المطلوبة بنفس المفهوم الذي يستخدمان به مع DOS ويجب وضعهما بين علامتي تنصيص فمثلا الوظيفة ADIR("*.DBF")تعطي عدد الملفات التي تنتهي بالاسم الممتد DBF.والموجودة على الدليل الحالي.

المثال الآتي يضع عدد الملفات الموجودة على الدليل الحالي داخل حقل ذاكرة اسمه NUM

```
NUM = ADIR("*.*")
```

فإذا قررت نقل معلومات عن الملفات إلى مصفوفات أخرى استخدم الاختيارات الخمسة التالية أو بعضها حسب رغبتك لأنها اختيارية. ويجب أن تُعرِّف كل مصفوفة داخل أمر DECLARE أو بأمر مستقل ويجب أن يكون عدد عناصر المصفوفات الخمسة المذكورة مطابقا لعدد الملفات الموجودة على الدليل.

وإليك اختيارات الوظيفة ADIR()ومعنى كل منها.

الاختيار	معناه
<array1>	اسم المصفوفة التي ستشتمل على أسماء الملفات الموجودة
<array2>	اسم المصفوفة التي ستشتمل على أحجام الملفات الموجودة
<array3>	اسم المصفوفة التي ستشتمل على تواريخ إنشاء الملفات الموجودة
<array4>	اسم المصفوفة التي ستشتمل على وقت إنشاء الملفات الموجودة
<array5>	اسم المصفوفة التي ستشتمل على حالة الملفات (Attribute)

إذا استخدمت الاختيار <array5>ضمن الوظيفة فإن المعلومات التي ستوضع داخل المصفوفة عن حالة الملفات هي:

الحالة (Attribute)	معناها
A	الملفات التي يمكن قراءتها وكتابتها (Archive)
D	دليل (Directory)
H	ملفات مخفية (Hidden)
R	ملفات قراءة فقط (Read only)
S	ملفات خاصة بنظام التشغيل (System)

المثال التالي امتداد للمثال السابق الذي نتج عنه وضع عدد الملفات داخل حقل ذاكرة اسمه NUM

```
DECLARE NAMES[NUM],SIZE[NUM],DATE[NUM],TIME[NUM]
ADIR("*.*",NAMES,SIZE,DATE,TIME)
```

وهو يشتمل على أمرين نوضحهما فيما يلي:

الأمر الأول ينشيء ٤ مصفوفات عدد عناصر كل منها مساو لعدد الملفات الموجودة.

الأمر الثاني يستخدم الوظيفة ()ADIR لنقل أسماء الملفات في المصفوفة الأولى (<array1>) وهي NAMES وأحجامها في المصفوفة الثانية (<array2>) وهي SIZE وتواريخ إنشائها في المصفوفة الثالثة (<array3>) وهي DATE ووقت إنشائها في المصفوفة الرابعة (<array4>) وهي TIME

وعندما تريد الحصول على معلومات عن مصفوفة واحدة أو مصفوفات معينة استخدم فراغات محل المصفوفة التي لا تريد معلومات عنها كما يتضح من المثال التالي:

```
EMPTY = " "
ADIR("*.*",EMPTY,EMPTY,DATE,EMPTY)
```

وفي هذه الحالة سيتم نقل تواريخ إنشاء الملفات فقط إلى المصفوفة DATE ويمكن تجاهل المصفوفة / المصفوفات إذا كانت في نهاية الأمر ولا تريد تعبئة محتوياتها فمثلا الأمر التالي يعطي نفس النتيجة.

```
ADIR("*.*",EMPTY,EMPTY,DATE)
```

تعبئة المصفوفة بمواصفات الحقول

تستخدم الوظيفة AFIELDS() لتحليل مواصفات ملف قاعدة البيانات وتخزين هذه المواصفات داخل مصفوفات فهي تملأ عناصر المصفوفة بالمعلومات التي تدل على مواصفات الحقول وهي اسم الحقل (Name) ونوعه (Type) وطوله (Len) وعدد الأرقام العشرية (Dec). ويجب تعريف اسم مصفوفة واحدة على الأقل داخل الوظيفة لتشتمل على أسماء الحقول فإذا أردت نقل معلومات وافية عن مواصفات الحقول يجب أن تشتمل الوظيفة على أسماء أربع مصفوفات تكون الأولى لاحتواء أسماء الحقول والثانية لأنواعها والثالثة لأطوالها والرابعة لعدد الأرقام العشرية. ويجب أن يتطابق عدد عناصر كل مصفوفة من هذه المصفوفات مع عدد حقول الملف. فإذا كان عدد عناصر المصفوفة أقل من عدد حقول الملف فإن الحقول الزائدة لن توضع معلوماتها داخل المصفوفة.

وهذه الوظيفة تشبه الوظيفة ADIR() التي شرحناها من قبل إلا أن هذه الوظيفة تضع معلومات عن حقول الملف داخل المصفوفات في حين تضع الوظيفة ADIR() معلومات عن الملفات الموجودة على الدليل. ولذلك فلابد من إنشاء المصفوفة قبل تعبئتها بمعلومات الحقول (الاسم والنوع والطول وعدد الأرقام العشرية) وبالتالي تحديد عدد عناصر المصفوفة.

المثال التالي عبارة عن برنامج صغير يظهر مواصفات الملف المفتوح.

```
* Program: STRU.PRG
USE STUDENTS
F_CNT = FCOUNT()      && ضع رقما يساوى عدد الحقول فى حقل الذاكرة F_CNT
* عرف المصفوفات التى ستشتمل على مواصفات الملف
DECLARE M_NAMES[F_CNT], M_TYPE[F_CNT], M_LEN[F_CNT], M_DEC[F_CNT]
* بامكانك استخدام امر واحد بدلا من الامرين السابقين هكذا
* DECLARE M_NAMES[FCOUNT()],M_TYPE[FCOUNT()], .... etc.
AFIELDS(M_NAMES, M_TYPE, M_LEN,M_LEN, M_DEC)
CNT = 1
* الاوامر التالية تنشىء دوارة لاظهار محتويات المصفوفات
FOR CNT = 1 TO F_CNT
  ? M_NAMES[CNT], M_TYPE[CNT], M_LEN[CNT], M_DEC[CNT]
NEXT
```

وفي هذا المثال استخدمنا الوظيفة ()FCOUNT لتخزين عدد حقول الملف في حقل ذاكرة ثم أنشأنا مصفوفات عدد عناصر كل منها مساويا لعدد حقول الملف. واستخـدمنـا الوظيفة ()AFIELDS لتخزين مواصفات الحقول داخل المصفوفات. وأخيرا أظهرنا محتويات المصفوفات.

إذا أردت الحصول على معلومات عن محتويات مصفوفة واحدة أو مصفوفات معينة استخدم فراغات محل المصفوفات التي لا تريد معلومات عنها هكذا:

```
EMPTY= " "
AFIELDS(M_NAMES,EMPTY,M_LEN)
```

في هذه الحالة ستحصل على أسماء الحقول وأطوالها فقط وكما تلاحظ تجاهلنا ذكر اسم المصفوفة M_Dec لأنها في آخر المجموعة.

الوظيفة ACHOICE( )

نظرا لأهمية هذه الوظيفة وصعوبتها بالنسبة للوظائف السابقة فسوف نتمهل في شرحها وسوف نعطي مثالا شاملا عبارة عن برنامج كامل يشرح استخدام هذه الوظيفة.

تستخدم هذه الوظيفة باختصار لاظهار قائمة تستخدم الشريط المضاء للانتقال بين اختياراتها. وتضع اختيارات هذه القائمة داخل مصفوفة. وسوف نبدأ بشرح شكل الوظيفة والاختيارات التي تشتمل عليها.

تأخذ الوظيفة ACHOICE() الشكل العام التالي:

ACHOICE(*<expN1>*, *<expN2>*, *<expN3>*, *<expN4>*, *<array1>*
[,*<array2>* [,*<expC>* [*<expN5>* [*<expN6>*]]]])

وإليك شرح الاختيارات الواردة بها:

<expN1>...<expN2> : تمثل مكان الركن اليسار العلوي للقائمة

<expN3> ..<expN4> : تمثل مكان الركن اليمين السفلي للقائمة

<array1> : المصفوفة التي تشتمل على اختيارات القائمة.

<array2> : مصفوفة اختيارية ـ تشتمل على القيمة المنطقية .T. أو .F. وتستخدم .T. للاشارة إلى أن العنصر الذي يخصها داخل المصفوفة رقم ١ يمكن اختياره بينما تستخدم .F. للاشارة إلى أن العنصر الذي يخصها داخل المصفوفة رقم ١ لا يمكن اختياره.

<expC> : اسم للوظيفة الخاصة التي توجه استخدام الوظيفة ACHOICE( ).

<expN5> : رقم الاختيار داخل القائمة الذي سيوضع عنده الشريط المضاء في بداية تشغيل الوظيفة.

<expN6> : رقم السطر الذي سيوضع عنده الشريط المضاء في بداية تشغيل الوظيفة منسوباً إلى النافذة التي تظهر فيها القائمة.

الشـــرح:

تستخدم الوظيفة ACHOICE( ) لانشاء قائمة تشتمل على اختيارات يتم الانتقال بينها باستخدام الشريط المضاء وتسمى هذه القائمة Pull-down menu .

وتظهر هذه القائمة داخل نافذة. وتوضع اختيارات القائمة داخل مصفوفة (<*ar-ray1*>) وتشبه الوظيفة ACHOICE( ) الأمر MENU TO... PROMPT..@

وتمتاز هذه الوظيفة على أمر MENU TO بميزتين :

الأولــى : أن أمر MENU TO لا يسمح بأكثر من ٣٢ اختيار داخل القائمة .

الثانية : إذا كان حجم النافذة التي تشتمل على الاختيارات أقل من عدد الاختيارات فإن القائمة تطوى لأعلى أو لأسفل تلقائياً.

عندما يتم اختيار واحد من الاختيارات الموجودة بالقائمة فإن الوظيفة ACHOICE( ) تشتمل على رقم العنصر المختار. فإذا ألغيت القائمة باستخدام Esc فإن الوظيفة ستشتمل على الرقم صفر. ويجوز أن تشتمل الوظيفة ACHOICE( ) على اسم وظيفة خاصة (User Defined Function) لكي توجه استخدام بعض المفاتيح أثناء التنفيذ. وفي هذه الحالة فإن المفاتيح ستخصص لها وظيفة معينة. فإذا لم تشتمل على وظيفة خاصة فستخصص للمفاتيح وظائف تلقائية تسمى (Default Mode) .

ويوضح الجدول التالي الوظائف المخصصة لبعض المفاتيح في حالة عدم استخدام وظيفة خاصة للتحكم في التنفيذ، وهو ما يطلق عليه Default Mode أي العمل بالطريقة التلقائية.

المفتاح	وظيفتـــه
↑	الانتقال إلى الاختيار السابق .
↓	الانتقال إلى الاختيار اللاحق .
Home	أول اختيار.
End	آخر اختيار.
PgUp	الانتقال إلى الصفحة السابقة (منسوبة إلى حجم النافذة) .

المفتاح	وظيفتـــــه
PgDn	الانتقال إلى الصفحة اللاحقة (منسوبة إلى حجم النافذة).
Ctrl-PgUp	أول اختيار.
Ctrl-PgDn	آخر اختيار.
Enter	تنفيذ الاختيار (يخصص رقم الاختيار للوظيفة (ACHOICE( )
Esc	إلغاء الاختيار (يخصص رقم صفر للوظيفة ACHOICE)
←	الغاء القائمة (يخصص رقم صفر للوظيفة ACHOICE)
→	الغاء القائمة (يخصص رقم صفر للوظيفة ACHOICE)
أول حرف	تنفيذ الاختيار الذي يبدأ بالحرف

بينما يوضح الجدول التالي الوظائف المخصصة لبعض المفاتيح في حالة استخدام وظيفة خاصة للتحكم في التنفيذ.

المفتاح	وظيفتـــــه
↑	الانتقال إلى الاختيار السابق.
↓	الانتقال إلى الاختيار اللاحق.
PgUp	الانتقال إلى الصفحة السابقة (منسوبة إلى حجم النافذة).

المفتاح	وظيفتـــه
PgDn	الانتقال إلى الصفحة اللاحقة (منسوبة إلى حجم النافذة) .
Ctrl-PgUp	الانتقال إلى أول اختيار.
Ctrl-PgDn	الانتقال إلى آخر اختيار.

وكما تلاحظ فإن مفاتيح :

Home – End – Enter – Esc

لاغية وغير مستخدمة .

وعندما يتم استدعاء وظيفة خاصة فإنها تتلقى ثلاث معطيات من الوظيفة ACHOICE( ) هي :

- حالة القائمة (Mode)
- رقم الاختيار الذي يقف عنده الشريط المضاء
- رقم السطر الذي يقف عنده الشريط المضاء منسوبا لبداية النافذة .

ويوضح الجدول التالي الحالات الخمس المتصلة بالوظيفة الخاصة ومعنى كل منها .

الحالة (Mode)	معنـاهـــا
0	لم يحصل اختيار.
1	محاولة نقل الشريط المضاء فوق أول اختيار.
2	محاولة نقل الشريط المضاء أسفل آخر اختيار.
3	حدوث خطأ (الضغط على مفتاح خطأ) .
4	كل عناصر القائمة لا يمكن اختيار.

وعندما تقرر استعمال وظيفة خاصة لتتحكم في تنفيذ اختيار القائمة فإن الـوظيفـة الخـاصبة ترسل قيما إلى الوظيفة ACHOICE( ) لتوجهها إلى ما يجب تنفيذه. ويوضح الجدول التالي هذه القيم والاجراء الذي ينفذ نتيجة تلقى ACHOICE( ) لها.

القيمة	الاجـــراء المطلوب
0	إلغـــــاء.
1	تنفيذ اختيـار.
2	التوقف عن التنفيذ.
3	توجيه الشريط المضاء إلى الاختيار التالي الذي يبدأ بآخر حرف أدخل.

مثـــال:

يوضح المثال التالي استخدام الوظيفة ACHOICE( ) في أبسط صورة

M_CHOICE = ACHOICE (03,05,08,12,MAIN)

وفي هذا المثال فإن 03 و05 تحدد الركن اليسار العلوي من القائمة بينما تحدد كلا من 08 و12 الركن اليمين السفلي للقائمة.

أما المثال التالي فيشتمل على برنامج يستخدم الوظيفة ACHOICE لتنفيذ قائمة ذات خمسة اختيارات.

```
* Program: ACH.PRG
CLEAR
*
DECLARE CHARY[5]        && انشيء مصفوفة ذات خمسة عناصر لتشتمل على اختيارات
                        && القائمة
* ضع اختيارات القائمة داخل عناصر المصفوفة *
CHARY[1] = "First  choice"
CHARY[2] = "Second choice"
CHARY[3] = "Third  choice"
CHARY[4] = "Forth  choice"
CHARY[5] = "Fifth  choice"
@ 8,4 TO 14,20 DOUBLE
* ضع قيمة الوظيفة فى حقل بالذاكرة ثم اظهر اختيارات القائمة *
MYCHOICE = ACHOICE(9,6,14,19,CHARY)
DO CASE
   CASE MYCHOICE = 0
        RETURN
   CASE MYCHOICE = 1
        DO PROCA
   CASE MYCHOICE = 2
        DO PROCB
   * <اوامر اخرى>
ENDCASE
*
PROCEDURE PROCA
@ 24,2 SAY "Good morning"
RETURN
*
PROCEDURE PROCB
@ 24,2 SAY "Good afretnoon"
RETURN
```

تـذكـر...!

التعامل مع المصفوفات مفهوم جديد لم يكن موجودا من قبل في قاعدة البيانات «دي بيس ثري بلاس». وقد تناولنا في هذا الفصل المفاهيم والوظائف التي تتعامل مع المصفوفات بدءا بإنشائها ثم تعبئة عناصرها سواء بقيم مختارة أو بمعلومات عن الملفات أو بمواصفات حقول الملف والبحث فيها وإدخال عنصر بين عناصر موجودة أو حذف عنصر موجود ونسخ محتويات مصفوفة إلى أخرى وترتيب عناصر المصفوفة.

وتم شرح الوظيفة ( )ACHOICE شرحا تفصيليا مع مثال يشتمل على برنامج يستخدم لاظهار قائمة اختيارات.

الفصل الثامن

مكتبة كلبر

Clipper Library

تشتمل مكتبة «كلبر» على العديد من البرامج التي تسهل إعداد نظم إدارة قواعد البيانات. مثل إنشاء ملفات قواعد البيانات وملفات الفهرسة وملفات التقارير والملصقات بالاضافة إلى العديد من برامج المساعدة الأخرى التي تخدم أغراض تعقب واكتشاف الأخطاء وأغراض إضافة برامج مكتوبة بلغات أخرى مثل «سي» أو «أسمبلي».

ويهدف هذا الفصل إلى توضيح كيفية استخدام البرامج الجاهزة (Utility programs) التي تأتي ضمن حزمة «كلبر».

المقصود بمكتبة «كلبر» البرامج والملفات التي تأتي ضمن حزمة «كلبر» والتي يمكن الاستفادة منها في إعداد نظم إدارة قواعد البيانات أو في بعض المساعدة التي لا غنى عنها لمستخدمي «كلبر» مثل تعقب واكتشاف أخطاء البرامج أو إنشاء وتصميم ملفات قواعد البيانات والفهرسة والتقارير والملصقات. . . الخ.

بعد إتمام تركيب «كلبر» على القرص الثابت انتقل إلى الدليل الخاص «بكلبر» واكتب أمر DIRثم اضغط مفتاح الادخال. ستحصل على قائمة بأسماء الملفات والبرامج التي ترد من الشركة المنتجة ويمكن تقسيم هذه الملفات والبرامج إلى ٣ مجموعات:

المجموعة الأولى: تشتمل على الملفات والبرامج التي تعتبر جزءا من قاعدة البيانات نفسها والتي تسمح بتطوير نظم إدارة قواعد البيانات.

المجموعة الثانية: برامج مكتوبة لتؤدي وظيفة محددة مثل تسهيل مهمة التعامل مع برامج مكتوبة بلغات «سي» أو «أسمبلي».

المجموعة الثالثة: برامج مساعدة (Utilities) تستخدم لأغراض إنشاء ملفات قواعد البيانات وملفات الفهرسة وملفات التقارير والملصقات لتكون بديلا لاستخدام شاشة المساعدة في «دي بيس ثري بلاس». وبديلا أيضا لبعض الأوامر التي لا تتعامل معها «كلبر» مثل أوامر:

CREATE/MODIFY - EDIT - BROWSE...

وسنورد فيما يلي بيانا مفصلا بالملفات والبرامج المساعدة في إعداد نظم إدارة قواعد البيانات والتي يطلق عليها Uility programsوكيفية استخدامها . وقد يكون استعراض كل هذه البرامج والملفات غير مطلوب الآن إلا أن الهدف من هذا الاتسعراض أن تكون على دراية بالامكانيات التي يمكن تحقيقها من مكتبة «كلبر». فمثلا قد لا تكون في حاجة الآن لاعداد ملف تقرير أو ملصقة باستخدام برامج المساعدة التي تأتي ضمن حزمة «كلبر» إلا أنك قد تحتاج لذلك فيما بعد. عندئذ يمكنك الرجوع إلى هذا الفصل لمعرفة كيفية تحقيق هذا الهدف.

استخدام البرامج الجاهزة

Utility Programs

قلنا إن حزمة «كلبر» تشتمل على مجموعة برامج جاهزة لتساعدك في إنشاء وتعديل الملفات وفي تطوير نظم إدارة قواعد البيانات ويطلق على هذه البرامج Utility programsوهذه البرامج هي:

١ ـ برنامج DBU.EXE

يسمح بإنشاء وإظهار وتعديل مواصفات الملفات بدون كتابة برامج أو استخدام مفسر وبإظهار محتويات الملفات وتعديلها والاضافة إليها أو الحذف منها بطريقة مشابهة لشاشة المساعدة في «دي بيس ثلاي بلاس».

٢ ـ برنامج RL.EXE

يستخدم هذا البرنامج لانشاء وتعديل ملفات التقارير (reports) والملصقات (labels).

٣ ـ برنامج LINE.EXE

يستخدم لاظهار و طباعة البرامج مع إمكانية ترقيم أوامرها.

٤ ـ برنامج INDEX.PRG

وهـو برنامج مكتوب «بكلبر» ليساعدك في إنشاء ملف NTX.وملف NTX. ملف فهرس أسرع من ملف NDX.الذي تستخدمه «دي بيس ثري بلاس».

٥ ـ برنامج DOT.EXE

يستخدم هذا البرنامج كبديل لنقطة توجيه الأوامر (dot-prompt) الموجودة في «دي بيس ثري بلاس».

وفيما يلي سنناقش بالتفصيل كيفية استخدام هذه البرامج.

استخدام برنامج DBU.EXE

يستخدم برنامج DBU.EXE لانشاء وتعديل مواصفات ملفات قاعدة البيانات (.DBF) أو ملف الفهرس (.NTX) ولتعديل وصيانة الملفات مثل إضافة سجلات أو حذف سجلات موجودة أو نسخ سجلات أو استبدال محتوياتها. . . الخ بطريقة مشابهة لاجراء هذه العمليات باستخدام شاشة المساعدة الموجودة في «دي بيس ثري بلاس».

وهذا البرنامج لا يأتي بالاسم DBU.EXE مع حزمة «كلبر» وإنما هو تجميع لثمانية ملفات تبدأ كلها بالحروف DBU. وتنتهي بالاسم الممتد PRG. (استخدم أمر DIR DBU*.PRG لتتعرف على أسمائها. ويتم ترجمتها وربطها معا للحصول على برنامج DBU.EXE ولكي تحصل على برنامج DBU.EXE يجب تنفيذ ملف MAKEDBU.BAT الذي يأتي ضمن حزمة كلبر هكذا:

C:\CLIPPER> MAKEDBU

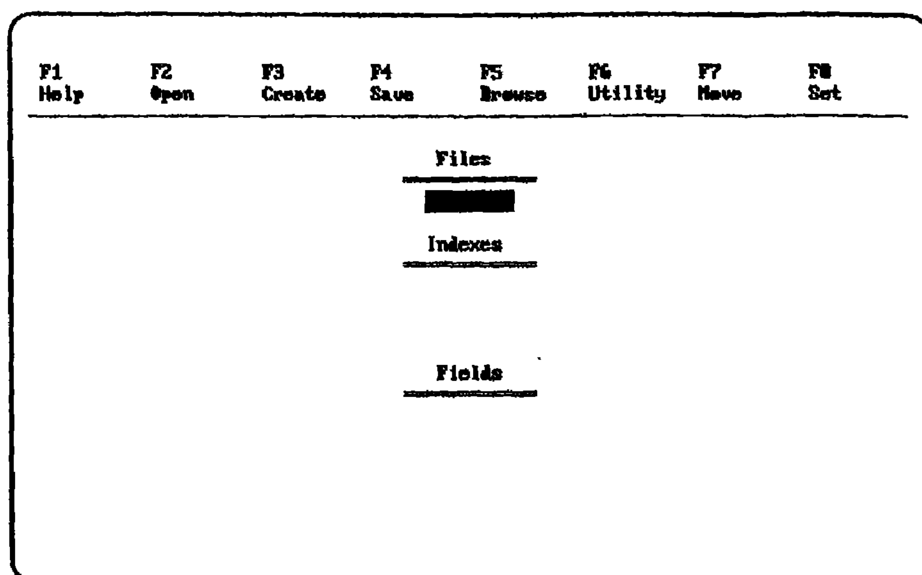
ملاحظة: ملف MAKEDBU.BAT يفترض أن كلبر موجودة تحت دليل اسمه Clipper.
فإذا كان اسم الدليل الموجود عندك يختلف عن هذا الاسم عدل ملف MAKEDBU.BAT بحيث يشتمل سطر PLINK86 على الاسم الصحيح للدليل.

وملف MAKEDBU.BAT يتولى نيابة عنك ترجمة وربط البرامج معا ويعطيك في النهاية برنامج DBU.EXE.

ولتشغيل البرنامج أدخل الأمر الآتي تحت محث DOS

C:\CLIPPER> DBU

ستحصل على شكل ٨ـ١ وهو عبارة عن قائمة تسمح بأداء جميع وظائف إنشاء وصيانة الملفات. وفي هذا الشكل تلاحظ أن السطر الأول يشتمل على أسماء مفاتيح الوظائف من F1 إلى F8 وتحت كل منها الوظيفة الأساسية التي يمكن الحصول عليها باستخدام هذا المفتاح. وبمجرد ضغط أحد المفاتيح ستحصل على قائمة منسدلة



شكل ١ـ٨

تشتمل على الاختيارات التي يمكن إجراؤها. وعموما يتم التعامل مع شكل ١ـ٨ بطريقة مشابهة للتعامل مع شاشة المساعدة الموجودة في «دي بيس ثري بلاس».

للخروج من هذه القائمة والعودة إلى نظام التشغيل اضغط مفتاح Esc ثم رد على رسالة Exit to Dos?(Y/N) باختيار Y.

وسنوضح فيما يلي الوظائف الأساسية التي يمكن أداؤها نتيجة استعمال مفاتيح الوظائف.

مفتاح *F1 (Help)*

يتسبب ضغط مفتاح F1 في فتح قائمة منسدلة تشتمل على اختيار واحد هو Help فإذا ضغطت مفتاح الادخال ستحصل على شاشة تشتمل على معلومات مساعدة عن كيفية التعامل مع البرنامج.

مفتاح *F2 (Open)*

نتيجة ضغط مفتاح F2 ستحصل على قائمة منسدلة بها ٣ اختيارات هي:

Database Index View

ومعناها ما هو نوع الملف الذي تريد فتحه من هذه الأنواع الثلاثة. ومعروف أن «كلبر» لا تتعامل مع ملفات VIEW بطريقة مباشرة لكنك تستطيع إنشاء هذه الملفات باستخدام برنامج DBU داخل النظام الذي تقوم بتطويره. وبذلك تتعامل مع ملفات VIEW. كما لو كنت تستخدم «دي بيس ثري بلاس».

مفتاح F3 (Create)

نتيجة لضغط مفتاح F3 ستحصل على قائمة تشتمل على اختيارين هما:

Database Index

ومعناها أي من هذين الملفين تريد إنشاءه. ويمكن استخدام هذه القائمة أيضا لتعديل مواصفات ملف موجود. لاحظ أن ملف Index لا يمكن إنشاؤه إلا إذا كان ملف Database مفتوحا. ولانشاء ملف قاعدة البيانات اختر Database وعندما تستحث لادخال اسم الملف اكتب اسم الملف ثم أدخل أسماء الحقول وأنواعها وأطوالها وعدد الأرقام العشرية بطريقة مشابهة لادخال مواصفات الملف في «دي بيس ثري بلاس» ثم اضغط مفتاح F4 واختر save (انظر شكل ٢-٨).

F1 Help | F2 Open | F3 Create | F4 Save | F5 Browse | F6 Utility | F7 Move | F8 Set

Files

Structure of <new file> Field 7

Field Name	Type	Width	Dec
STUDENTNO	Character	3	
FIRSTNAME	Character	12	
LASTNAME	Character	12	
ADDRESS	Character	20	
CITY	Character	10	
BIRTHDATE	Date	8	
SAUDI		1	

شكل ٢-٨

مفتاح F4 (Save)

نتيجة لضغط مفتاح F4 ستحصل على قائمة بها اختياران هما :

View struct

ومعناها أيا من هذين النوعين تريد حفظه على القرص الممغنط وكلمة struct تعني Structureأي مواصفات الملف الذي أنشأته من خلال قائمة Create.

مفتاح F5 (Browse)

نتيجة لضغط مفتاح F5ستحصل على قائمة بها اختياران هما :

Database View

ومعناها أي من هذين النوعين تريد إظهاره على الشاشة وتعديل محتوياته .

مفتاح F6 (Utility)

يظهر هذا المفتاح قائمة منسدلة بها ٦ اختيارات هي :

Copy Append Replace Pack Zap Run

وهي اختيارات النسخ والاضافة والاستبدال والحذف أما الاختيار Runفمعناه الخروج إلى نظام التشغيل وإصدار أي أمر مقبول من أوامر DOSوللعودة إلى برنامج DBUاضغط مفتاح Esc.

مفتاح F7 (Move)

يظهر هذا المفتاح قائمة منسدلة بها أربعة اختيارات هي

Seek Goto Locate Skip

وكلها اختيارات تسمح بتحريك المؤشر داخل الملف إلى سجل معين .

مفتاح F8 (Set)

يسمح هذا المفتاح بظهور قائمة منسدلة بها ٣ اختيارات هي :

Relation Filter Fields

وهذه الاختيارات تتيح ربط ملفين معا طبقا لشرط أو شروط معينة واختيار حقول من كلا الملفين في الملف الجديد .

استخدام برنامج RL.EXE

برنامج RL.EXE بديل لأمر CREATE/MODIFY LABEL الموجود في «دي بيس ثري بلاس» فهو يمكنك من إنشاء ملف تقارير (.FRM) أو ملف ملصقات (.LBL) أو تعديل أحدهما إذا كان موجودا من قبل. وكلمة RL اختصار لعبارة Report/Label.

وبمجرد إنشاء ملف تقرير (FRM). أو ملصقة (.LBL) باستخدام هذا البرنامج يمكن استخراج التقرير أو الملصقات أو طباعتها فيما بعد باستخدام أمر REPORT FORM في حالة التقارير أو أمر LABEL FORM في حالة الملصقة.

وهذا البرنامج لا يأتي بالاسم RL.EXE مع حزمة «كلبر» وإنما هو تجميع للملفات الثلاثة التالية:

RLBACK.PRG - RLDIALOG.PRG - RLFRONT.PRG

ويتم ترجمة هذه البرامج الثلاثة وربطها مع بعضها للحصول على برنامج RL.EXE. ولكي تحصل على برنامج RL.EXE يجب تنفيذ ملف MAKERL.BAT الذي يأتي ضمن حزمة «كلبر» هكذا:

```
C:\CLIPPER> MAKERL
```

ملاحظة: ملف MAKERL.BAT يفترض أن «كلبر» موجودة تحت دليل اسمه Clipper.
فإذا كان اسم الدليل الموجود عندك يختلف عن هذا الاسم عدل ملف MAKERL.BAT
بحيث يشتمل سطر Plink86 على الاسم الصحيح للدليل.

ويتولى ملف MAKERL.BAT نيابة عنك ترجمة وربط البرامج الثلاثة معا ويعطيك في النهاية برنامج RL.EXE. ولتشغيل البرنامج أدخل الأمر التالي تحت محث DOS

```
C:CLIPPER> RL
```

ستحصل على شكل ٣ـ٨ وهو عبارة عن قائمة أفقية بها ٣ اختيارات هي :

Quit　　　　Label　　　　Report

فإذا كنت في حاجة لانشاء أو تعديل ملف تقرير اختر Report

وإذا كنت في حاجة لانشاء أو تعديل ملف ملصقات اختر Label

وعند الرغبة في الخروج إلى نظام التشغيل اختر Quit

إنشاء تقرير بواسطة برنامج RL.EXE

فبفرض أننا نريد إنشاء ملف تقرير. اختر Report ستحصل على شكل ٤ـ٨ وهو شكل يسمح لك بإدخال اسم التقرير. ورغبة في زيادة الايضاح سنستخدم مع الشرح مثالا ينشىء ملف تقرير باسم REPORT1.FRM اكتب REPORT1 واضغط مفتاح الادخال. ثم اختر ok (أو اختر Cancel إذا كنت تريد الخروج من البرنامج).

بعد اختيار اسم للملف ستحصل على شكل ٥ـ٨ وهذا الشكل يسمح لك بتوصيف الحقول. ويشتمل السطر الأول من شكل ٥ـ٨ على أسماء مفاتيح الوظائف من F1 إلى F7 و F10 وتحت كل منها الوظيفة التي يسمح هذا المفتاح بالقيام بها.

Report　Label　Quit

شكل ٣ـ٨

```
Report   Label   Quit

        Enter a filename

        File

               Ok      Cancel
```

شكل ٤ـ٨

```
F1     F2      F3      F4      F5      F6      F7     F10
Help   Layout  Groups  Fields  Delete  Insert  Go To  Exit

                                              File REPORT1.FRM
                                                       Field 1
                  == Field Definitions ==              Total 1

Contents

Heading
       1
       2
       3
       4

Formatting
Width      10
Decimals   0
Totals     N
```

شكل ٥ـ٨

وعليك أن تضغط على المفتاح المناسب للاختيار الذي تريده. ولادخال التوصيف المطلوب اكتب اسم الحقل أمام Contents ثم اضغط مفتاح الادخال سينتقل المؤشر تلقائيا إلى Heading اكتب العنوان المختار لهذا الحقل حتى مدى ٤ سطور ثم أكمل باقي المواصفات وهي طول الحقل Width عدد الأرقام العشرية (Decimals) وهل ترغب في تجميعه أم لا (Totals) وأمامها قيمة تلقائية N وتعني لا ويمكنك تغييرها إلى Y . ونوضح فيما يلي الوظائف التي يمكن أداؤها نتيجة استعمال مفاتيح الوظائف.

مفتاح F1 (Help)

يظهر شاشة معلومات مساعدة.

مفتاح F2 (Layout)

يظهر شاشة تسمح باختيار عنوان الصفحة (Page header) في حدود ٤ سطور بالاضافة إلى مجموعة اختيارات تتحكم في شكل صفحة التقرير وهي عرض الصفحة (Page width) ، بداية الهامش الأيسر (Left margin) ، نهاية الهامش الأيمن (Right margin) عدد سطور الصفحة (Line per page) والمسافة بين سطور التقرير واختيار قفز صفحة قبل أو بعد طباعة واختيار إظهار رقم الصفحة والتاريخ في البداية أولا (plain page). (انظر شكل ٦ـ٨).

وهذه الاختيارات مشابهة تماما لاختيارات قائمة Options التي تظهر عند تصميم التقرير باستخدام «دي بيس ثري بلاس».

مفتاح F3 (Groups)

يظهر شاشة تشتمل على اختيارات مشابهة لاختيارات قائمة Groups الموجودة في تصميم التقرير باستخدام «دي بيس ثري بلاس» وفي هذه الشاشة يجب تحديد اسم الحقل الذي سيقسم التقرير إلى مجموعات بناء على بياناته. Group on expression وهذه الشاشة تعطيك الفرصة لتحديد هذا الحقل والعنوان الذي سيسبق كل مجموعة (Group Heading) واختيار هل تريد تقرير مختصر أم لا (Summary Report Only) واختيار بداية كل مجموعة من صفحة جديدة أم لا (Page Eject After Group)

```
F1      F2      F3      F4      F5      F6      F7      F10
Help    ...     Groups  Fields  ...     ...     ...     Exit
                                                    File REPORT1.FRM
                       == Page Header ==

                Formatting
                PageWidth         80
                LeftMargin        8
                RightMargin       0
                Lines Per Page    58
                Double Spaced?    N

             Printer Directives
             Page Eject Before Print   Y
             Page Eject After Print    N
             Plain Page                N
```

شكل ٦ـ٨

وفي الجزء الأخير من الشاشة إمكانية تقسيم المجموعات إلى مجموعات فرعية (Sub-Group on expression) واختيار عنوان للمجموعات الفرعية (انظر شكل ٧ـ٨) وفي شكل ٧ـ٨ اخترنا أن يقسم التقرير إلى مجموعات بحيث تشتمل كل مجموعة على بيانات شركة (Company) وأن تبدأ بيانات هذه الشركة من صفحة جديدة. وتطبع كل السجلات. ولسنا في حاجة لتقسيم التقرير إلى مجموعات فرعية.

مفتاح F4 (Fields)

وهـذه الشـاشـة تظهر تلقائا بمجرد اختيار اسم الملف كما يمكنك إظهارها باستخدام مفتاح F4 (وقد شرحنا مكونات هذه الشاشة منذ قليل). وفيها يتم إدخال أسماء الحقول أو التعبيرات التي ستنتج أعمدة التقرير. فمثلا التعبير التالي معقول لدى «كلبر».

```
"MR."+LTRIM(FIRSTNAME)+" "+LASTNAME
```

ويتم الانتقـال في هذه الشـاشـة من الحقل السابق إلى الحقل اللاحق أو العكس باستخدام مفتاحي PgUp أو PgDn (انظر شكل ٨ـ٨).

```
F1      F2      F3      F4      F5      F6      F7      F10
Help    Layout  ...     Fields  ...     ...     ...     Exit
------------------------------------------------------------
                                            File REPORT1.FRM
                 === Group Specifications ===

Group On Expression      company
Group Heading            Compny name

Summary Report Only      N
PageEject After Group    N

               === Sub-Group Specifications ===

Sub-Group On Expression
Sub-Group Heading
```

شكل ٨ـ٧

```
F1      F2      F3      F4      F5      F6      F7      F10
Help    Layout  Groups  Fields  Delete  Insert  Go To   Exit
------------------------------------------------------------
                                            File REPORT1.FRM
                                                     Field 1
                 === Field Definitions ===           Total 2

Contents "Mr. "+LTRIM(FIRSTNAME)+" "+LASTNAME

Heading
      1  Student's
      2  name
      3
      4

Formatting
Width     30
Decimals   0
Totals    N
```

شكل ٨ـ٨

مفتاح F5 (Delete)

يسمح هذا المفتاح بحذف الحقل الموجود على شاشة الحقول (Fields) ولذلك فإن الضغط عليه لا يصلح إلا إذا كانت قائمة الحقول ظاهرة. ويتم حذف الحقل من أعمدة التقرير.

مفتاح F6 (Insert)

يسمح بإدخال حقل بين حقول موجودة ويتم إدخال الحقل قبل الحقل الظاهر على الشاشة.

مفتاح F7 (Go to)

للانتقال من حقل إلى آخر داخل أعمدة التقرير وذلك بالاضافة لامكانية استخدام مفتاح PgUp للانتقال إلى الحقل السابق ومفتاح PgDn للانتقال إلى الحقل اللاحق ولذلك فبعد ضغط هذا المفتاح تستحث دائما لادخال رقم الحقل المراد الذهاب إليه داخل التقرير.

مفتاح F10 (Exit)

ضغط هذا المفتاح يتيح ٣ اختيارات هي

OK: للخروج من حفظ التقرير على ملف بالاسم المختار واسم ممتد FRM.

NO: بدون حفظ التقرير أو التعديلات التي تمت.

Cancel: ومعناها إلغاء قرار الخروج والعودة لاستئناف العمل في شاشة تصميم التقرير وفي حالة اختيار OK أو NO سترجع إلى الشاشة الأولى لاختيار تقرير أو ملصقة. وننصح باختيار OK إذا كنت قد صممت تقريرا.

إنشاء ملف ملصقات

إذا اخترت Label من القائمة الأولى (راجع شكل ٣-٨) فستحصل على شاشة تتيح لك تصميم شكل الملصقة واختيار عناصرها وهي قريبة جدا من شاشة تصميم الملصقة التي تستخدمها «دي بيس ثري بلاس».

وبمجرد اختيار Label ستحصل على شاشة لادخال اسم ملف الملصقة أو اختيار ملف موجود لتعديله. ولأننا نهدف إلى التدريب العملي أثناء الشرح اكتب

ملف STLBL. وستفهم كلبر أنـك تريد إنشاء ملف باسم STLBL. اختر OK ثم اضغط مفتاح الادخال (إذا أردت الرجوع للقائمة السابقة اختر Cancel).

ستحصل على شكل ٩ـ٨ وهو الشكل الذي يتيح لك تصميم شكل الملصقة وتجد في السطر الأول أسماء مفاتيح الوظائف التي يمكن استخدامها ووظيفة كل منها وإليك بيان تفصيلي بهذه المفاتيح.

مفتاح F1 (Help)

للحصول على شاشة معلومات مساعدة.

مفتاح F2 (Toggle)

يسمح بالتديل بين اختيار شكل الحقل واختيار حقل جديد.

مفتاح F3 (Formats)

يسمح باختيار حجم الملصقة (من ٥ أحجام متاحة) (انظر شكل ١٠ـ٨).

```
F1      F2       F3                                          F10
Help    Toggle   Formats                                     Exit
-----------------------------------------------------------------
                                                  File STLBL.LBL
Dimensions                  Formatting
      Width    35                 Margin    0
      Height   5                  Lines     1
      Across   1                  Spaces    0

      Remarks  

          CONTENTS
Line 1=====================================================
```

شكل ٩ـ٨

```
F1        F2        F3                                          F10
Help      Toggle    Formats                                     Exit
                                                      File STLBL.LBL
 Dimensions               Formatting
      Width    35              Margin       0
      Height    5              Lines        1
      Across    1              Spaces       0

      Remarks

             CONTE
Line 1                 3 1/2 x 15/16 by 1
                       3 1/2 x 15/16 by 2
                       3 1/2 x 15/16 by 3
                       4 x 17/16 by 1
                       3 2/10 x 11/12 by 3 (Cheshire)
```

شكل ١٠-٨

مفتاح F10 (Exit)

يسمح بالخروج من حفظ التعديلات أو إلغائها.

لاحظ أن علامة , (comma) في السطر تعني نهاية السطر. فإذا اضطررت لاستخدام هذه العلامة كفاصل داخل عنوان مثلا فيجب وضعها بين علامتي التنصيص "" لتفهم «كلبر» أنها ليست نهاية السطر.

استخدام برنامج INDEX

يأتي مع حزمة «كلبر» برنامج بالاسم INDEX.PRG وهو عبارة عن برنامج مكتوب «بكلبر» لغرض مساعدتك في كتابة برامج مماثلة ولذلك يجب ترجمة هذا البرنامج وربطه قبل استخدامه لاستخراج ملف INDEX.EXE وهو الذي يستخدم فيما بعد:

ملاحظة: راجع الفصل السادس لتعرف كيفية ترجمة وربط البرنامج لاستخراج صورة جاهزة للتنفيذ (.EXE) منه.

ولكي تستخدم برنامج INDEX.EXE استخدمه بهذا الشكل

INDEX <d: filename>

حيث :d اسم مشغل القرص ويمكن تجاهلها في حالة وجود الملف على الدليل الحالي و filename اسم ملف قاعـدة البيانـات المطلوب فهرسته. ولا يلزم إضافة الاسم الممتد DBF. لاسم الملف لأن «كلبر» ستفهم أن هذا ملف قاعدة بيانات.

بعد ذلك ستظهر أمامك حقول الملف وستسألك كلبر عن اسم ملف الفهرس (Index filename) واسم الحقل/الحقول التي ترغب في ترتيب أو فهرسة الملف طبقا لمحتوياتها.

استخدام برنامج LINE.EXE

برنامج LINE.EXE يأتي ضمن حزمة «كلبر» في صورته الجاهزة للتنفيذ بعكس البرامج التي شرحناها سابقا.

ويستخدم هذا البرنامج لاعطاء أرقام مسلسلة للسطور الموجودة في البرنامج أثناء طباعة البرنامج على الطابعة أو إظهاره على الشاشة كما يسمح بإظهار أو طباعة جزء من البرنامج فقط وترقيم السطور المطلوبة.

فمثلا إذا أردت إظهار محتويات برنامج اسمه MYPROG.PRG على الشاشة فإنك تستخدم أمر TYPE وهو أحد أوامر DOS بالشكل الآتي:

TYPE MYPROG.PRG

وفي كلتـا الحالتين ستظهر سطور البرنامج بدون أرقام مسلسلة. فإذا أردت إظهار محتويات البرنامج مع إعطاء أرقام مسلسلة لسطور البرنامج استخدم برنامج LINE.EXE بالشكل الآتي:

LINE MYPROG.PRG

(ولابد من كتابة الاسم الممتد لأنه برنامج LINE.EXE يظهر أي ملف مكتوب بشفرة ASCII مثل أمر TYPE تماما).

وإذا أردت توجيه المخرجات إلى الطابعة مرقمة أعد استخدام الأمر هكذا:

```
LINE MYPROG.PRG >PRN
```

ويمكن إضافة رقم للشكل السابق للتحكم في إظهار أو طباعة جزء محدد من البرنامج وعند الطباعة يتم طباعة ٥ سطور قبل الرقم الذي حددته و١٠ بعده فمثلا الشكل التالي سيظهر لنا السطور من ٤٥ إلى ٦٠.

```
LINE MYPROG.PRG 50
```

استخدام برنامج DOT.EXE

برنامج DOT.PRG يأتي ضمن حزمة «كلبر» وبعد ترجمته وربطه مع نظام التشغيل واستخراج برنامج DOT.EXE يمكن استخدامه كبديل لنقطة توجيه الأوامر الموجودة في «دي بيس ثري بلاس» ويمكن الاستفادة من هذا البرنامج بعد تجهيزه في الصورة الجاهزة للتنفيذ (.EXE) في مراجعة قواعد كتابة الأوامر أو للحصول على بعض النتائج المباشرة بدون انتظار لترجمة وربط البرامج بنفس الطريقة التي نستخدم بها أوامر «دي بيس ثري بلاس».

بعد ترجمة وربط البرنامج أدخل الآن الآتي تحت محث DOS لاستدعاء البرنامج

```
C:\CLIPPER> DOT
```

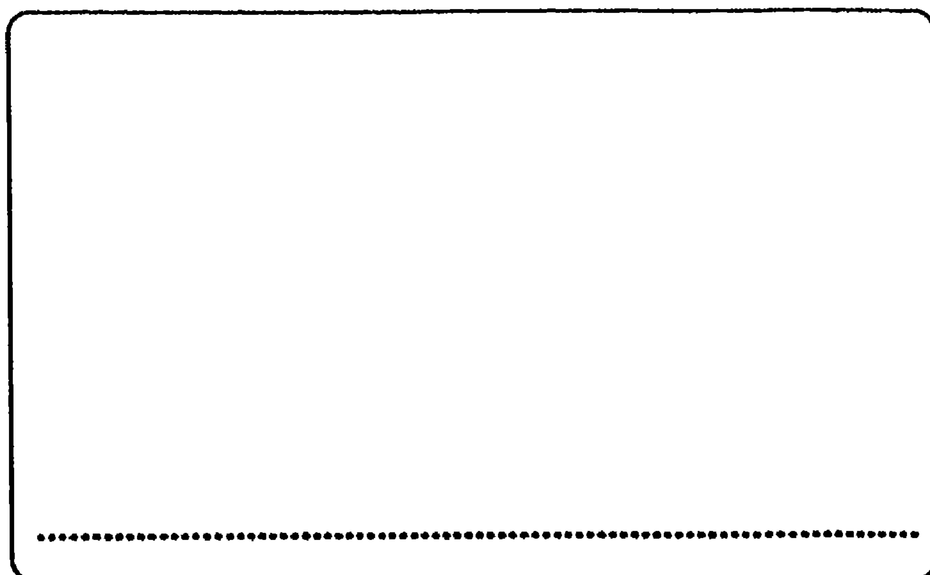
بعد ضغط مفتاح الادخال ستحصل على شكل ١١ـ٨ لاحظ مؤشر الشاشة أسفل السطر المنقط ليبين لك مكان إدخال الأوامر ولاحظ المربع الصغير الموجود في الركن اليسار العلوي من الشاشة ليحدد مكان ظهور ناتج الأوامر.

اكتب أمر DIR ثم اضغط مفتاح الادخال ستحصل على شكل ١٢ـ٨.

ويمكن الاستفادة من هذا البرنامج في معرفة الأوامر الممنوعة في كلبر وذلك لأن «كبر» ستعطيك الرسالة التالية في حالة إدخال أمر خطأ أو غير موجود بها.

```
Unrecognized Command, F1 for Help
```

للخروج من البرنامج والعودة إلى نظام التشغيل اكتب أمر QUIT.



شكل ١١ـ٨

Database Files	# Records	Last Update	Size
TEMPSCAT.DBF	0	12/04/89	163
SCATEG.DBF	0	12/04/89	163
CATEG.DBF	2	09/21/90	257
PRODUCT.DBF	0	12/04/89	354

شكل ١٢ـ٨

تـذكـر...!

تنـاولنـا في هذا الفصـل كيفية استخدام أشهر البرامج التي تأتي ضمن حزمة «كلبر». والتي تسهل إعداد نظم إدارة قواعد البيانات وهذه البرامج يطلق عليها عبارة Utility Programs أي برامج الخدمات وذلك لأنها تسهل التعامل مع ملفات قاعدة البيانات وملفات الفهرسة وتساعد في تصميم ملفات التقارير والملصقات. . . الخ .

وتأتي جميع هذه البرامج والملفات ضمن حزمة «كلبر» بعضها في صورته المصدرية وبعضهـا مترجم في صورة برامـج هدف (.OBJ) ولتحقيق أقصى استفـادة من هذه البرامج والملفات يلزم ترجمة بعضها أو ربط تلك التي تمت ترجمتها معا .

الفصل التاسع

تعقب وتصحيح أخطاء البرامج Clipper Debugger

مهما كانت خبرتك وقدرتك على تطوير نظم إدارة قواعد البيانات، فلابد من الوقوع في أخطاء، إذ اقتضت حكمة السماء أن يبقى الكمال لله وحده والعصمة لأنبيائه صلوات الله وسلامه عليهم جميعا.

ويشرح هذا الفصل كيفية التعامل مع مكتشف الأخطاء الذي يأتي ضمن حزمة «كلبر» ـ ويسمى Clipper Debugger ـ لتسهيل تعقب الأخطاء وتصحيحها. ويبدأ الفصل بشرح الاصطلاح Bug ثم يشرح كيفية ربط مكتشف الأخطاء ضمن النظام أو البرنامج الذي تنوي تجربته. وأخيرا كيفية استدعاء واستخدام مكتشف الأخطاء.

من البديهيات المسملة أن المبرمج مهما كان ضليعا وماهرا فلابد أن يقع في أخطاء. ولذلك فإن عملية اكتشاف الأخطاء وتصحيحها قبل تجربة النظام أو تسليمه للعميل لا تقل في أهميتها عن كتابة البرنامج أو النظام نفسه.

وكانت عملية اكتشاف وتصحيح الأخطاء بالنسبة لمبرمجي «دي بيس ثري بلاس» تتم باستخدام المفسر (Interpretor). وهي عملية سهلة وتلقائية. أما في «كلبر» فإن الأمر يتطلب استخدام برنامج موجود ضمن برامج «كلبر» ليتولى تسهيل هذه المهمة. وننصح الذين يستخدمون برامج مكتوبة «بدي بيس ثري بلاس» ولا تشتمل على أوامر «كلبر» التي لا تستخدمها «دي بيس» ننصحهم باستخدام المفسر الموجود في «دي بيس ثلاي بلاس» لأنه أسهل بكثير ولا يتطلب جهدا إضافيا. إذ يكفي تشغيل البرنامج، فإذا حصلت على رسالة خطأ أو حصلت على نتائج غير المتوقعة اضغط مفتاح Esc لالغاء تنفيذ البرنامج ثم اذهب لتصحيح الأخطاء ثم أعد تشغيل البرنامج.

وتسمى أخطاء البرنامج بلغة الحاسب "Bugs" وتسمى عملية تعقب الأخطاء وتنقيحها Debugging ويسمى البرنامج الذي يتولى تعقب الأخطاء وتنقيحها Debugger.

وتشتمل حزمة «كلبر» على برنامج يساعد في عملية اكتشاف وتعقب الأخطاء يسمى Clipper Debugger. وسنطلق عليه في هذا الفصل مكتشف الأخطاء. واسم هذا البرنامج هو DEBUG.OBJ.

ويستخدم مكتشف الأخطاء القوائم الأفقية والقوائم المنسدلة عنها لتسهيل عملية اكتشاف أخطاء البرنامج وتشتمل هذه القوائم على العديد من الاختيارات التي تعطي معلومات مفيدة عن البرنامج أثناء تنفيذه والتي تسهل عملية اكتشاف الأخطاء.

وتتلخص الوظائف الأساسية لبرنامج DEBUG.OBJ فيما يلي:

١ ـ إمكانية توقف التنفيذ بعد كل أمر موجود في البرنامج.

٢ ـ إمكانية توقف التنفيذ واستدعاء مكتشف الأخطاء في أماكن محددة داخل

البرنامج تصل إلى ١٦ موضعا وتسمى Breakpoints.

٣ ـ إمكانية إظهار التعبيرات التي يشتمل عليها البرنامج .

٤ ـ إمكـانية إظهـار قائمـة تسمى Watch box لتعـطي معلومات عن اسم البرنامج الذي ينفذ ورقم السطر وإمكانية توقيف البرنامج من عدمها .

ولكي تتأكد أن هذا البرنامج موجودا عندك اكتب الأمر الآتي تحت محث نظام التشغيل بعد الانتقال إلى الدليل الذي يشتمل على برنامج Clipper

```
C:\CLIPPER> DIR DEBUG.OBJ
```

ربط مكتشف الأخطاء ضمن النظام
Linking the Debugger

يجب ربط برنـامج DEBUG.OBJ مع النظام الذي تنوي تشغيله باستخدام مكتشف الأخطاء لتستخرج برنامجا جاهزا للتنفيذ (.EXE) وهنا يجب الاشارة إلى نقطة هامة وهي ضرورة ترجمة البرنامج أو النظام الذي سيتم ربطه مع برنامج DEBUG.OBJ بدون استخدام الاختيار I- (ومعناها كما أوضحنا في الفصل السادس ترجمة البرنامج بدون إضافة رقم السطر) فإذا رغبت في إضافة الاختيار I- إلى مترجم «كلبر» فيجب أن يتم بعد الانتهاء من اكتشاف وتصحيح الأخطاء ولربط برنامج أو نظام مع مكتشف الأخـطاء أضف إلى عبـارة FILE برنـامج DEBUG.OBJ بعـد البرنامج / البرامج التي تريد ربطها. (راجع الفصل السادس).

ونوضح فيما يلي مثالا لربط برنامج سبق ترجمته باسم MENU.OBJ مع مكتشف الأخطاء DEBUG.OBJ والمثال يفترض أن كلا من: «كلبر» ومكتبة «كلبر» وبرنامج MENU.OBJ وبرنامج DEBUG.OBJ على دليل واحد.

Plink86 FI MENU, DEBUG LIB CLIPPER

هام: *يجب ذكر اسم البرنامج الذي تنوي اكتشاف أخطائه قبل برنامج DEBUG.OBJ*

استخدام مكتشف الأخطاء

قبل استخدام مكتشف الأخطاء يجب أولا استدعائه للتنفيذ ويتم استدعاء مكتشف الأخطاء بإحدى ثلاث طرق:

١ - الضغط على مفتاح Alt-D أثناء تنفيذ البرنامج.

٢ - إذا اكتشف البرنـامج الـوظيفة ()ALTD وهذه الوظيفة تستخدم داخل الـبرنـامج لاستدعاء مكتشف الأخطاء (راجع وظائف «كلبر» في الباب الثالث من هذا الكتاب).

٣ ـ عندما يصل البرنامج إلى إحدى النقاط التي تتطلب توقف التنفيذ واستدعاء مكتشف الأخطاء وتسمى Breakpoints.

وعند استدعاء مكتشف الأخطاء بواحدة من هذه الطرق الثلاث يظهر في أعلى الشاشة التي أمامك قوائم واختيارات مكتشف الأخطاء. ففي السطر الأول من الشاشة تظهر قائمة أفقية تشتمل على ستة اختيارات هي :

Control Display Variable Help Break Watch

ويتبع كل اختيار منها قائمة منسدلة تشتمل على عدة اختيارات أخرى (سنناقشها بالتفصيل بعد قليل). وكذلك يظهر في الركن اليمين العلوي قائمة تشتمل على اسم البرنامج الذي ينفذ ورقم السطر وإمكانية توقف البرنامج من عدمها (Breakpoint) (انظر شكل ١ـ٩). ويتم الانتقال بين القوائم الأفقية باستخدام مفاتيح الأسهم ← أو → أما الانتقال داخل القوائم المنسدلة فيتم باستخدام مفاتيح الأسهم ↓ أو ↑ لوضع الشريط المضاء على الاختيار المطلوب ثم ضغط مفتاح الادخال.

```
Control  Display  Variable  Help  Break  Watch
Go                Add        Edit        Delete      Retu   Proc SACUST
Go (animation)                                              Line      19
Go (key)                                                    Break points <off>
Single Step
DOS Shell
Break Toggle              Customer main data
Quit
            Account No.   :
            Customer name:
            Company Name  :
            Address       :
            City          :
            Phone         :
```

شكل ١ـ٩

ويوضح الجدول التالي المفاتيح التي تستخدم مع مكتشف الأخطاء ووظيفة كل منها.

المفتاح	وظيفتــــــه
← أو →	الانتقال بين اختيارات القائمة الأفقية حسب اتجاه السهم
↓ أو ↑	الانتقال بين اختيارات القائمة المنسدلة حسب اتجاه السهم
Home	الانتقال إلى أول اختيار في القائمة المنسدلة
End	الانتقال إلى آخر اختيار في القائمة المنسدلة
PgUp	الانتقال إلى الصفحة السابقة
PgUn	الانتقال إلى الصفحة اللاحقة
Ctrl-PgUp	الانتقال إلى أول صفحة
Ctrl-PgDn	الانتقال إلى آخر صفحة

كما توجد مجموعة أخرى من المفاتيح تقوم بوظائف أخرى مع مكتشف الأخطاء تسمى Speed keys سنناقشها بعد قليل عند مناقشة اختيارات القوائم.

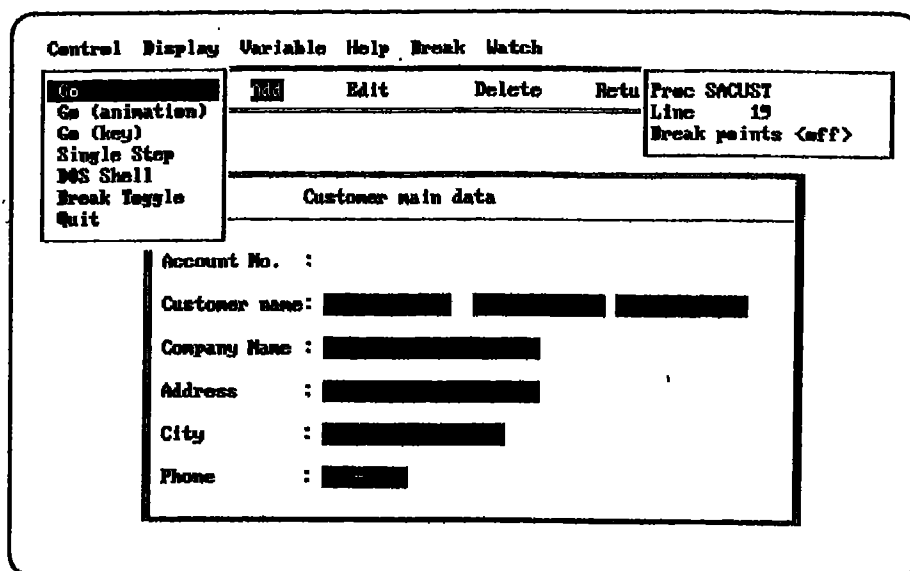
وسنناقش فيما يلي الاختيارات الستة الرئيسية التي تظهر عند استدعاء مكتشف الأخطاء.

١) قائمة *Control*

يتم فتح هذه القائمة تلقائيا عند استدعاء مكتشف الأخطاء. ويوضح شكل ٩ـ٢ هذه القائمة والاختيارات السبعة التي تشتمل عليها وفيما يلي نوضح المقصود بكل من هذه الاختيارات:

١ ـ الاختيار Go

يستخدم لتنفيذ البرنامج بدون مكتشف الأخطاء حتى يتم استدعاؤه مرة أخرى أو يصل البرنامج إلى إحدى نقاط التوقف (Breakpoints).



شكل ٢-٩

٢ - الاختيار Go (animation)

يشبـه الاختيار السابق إلا أنه يظهر معلومات عن البرنامج داخل مستطيل يسمى Watch box أثناء تنفيذ البرنامج ولأن هذه المعلومات تظهر بسرعة أثناء تنفيذ الـبرنامج لدرجة أنك لن تستطيع متابعتها فيفضل وضع الوظيفة INKEY() كأحد التعبـيرات التي تكتب مع الاختيار Watch Box. فمثلا INKEY(.1) يخصص عشر الثانية لكل سطر لتتمكن من قراءته.

٣ - الاختيار Go (key)

يتطلب هذا الاختيار ضغط أحد المفاتيح قبل العودة إلى تنفيذ البرنامج. وهو يشبه الاختيارين السابقين.

٤ - الاختيار Single Step

يتسبب هذا الاختيار في تنفيذ الـبرنامج سطرا سطرا. ويتطلب ضغط أحد المفاتيح قبل تنفيذ كل سطر. وهذا يمكنك من متابعة مخرجات البرنامج أثناء التنفيذ

وملاحظة ترتيب تنفيذ الأوامر داخل البرنامج .

٥ ـ الاختيار Dos Shell

يتسبب هذا الاختيار في استدعاء نظام التشغيل DOS أثناء تنفيذ كل من البرنامج ومكتشف الأخطاء . وقد يلزمك هذا الاجراء لأداء بعض المهام التي تتطلب «دوس» مثل إضافة ملف أو تشكيل قرص جديد . وللعودة إلى مكتشف الأخطاء مرة أخرى حيث كنت تقف اكتب Exit تحت محث نظام التشغيل .

٦ـ الاختيار Break Toggle

يسمح هذا الاختيار بالتبديل بين الوضع ON والوضع OFF لنقاط التوقف المحددة داخل البرنامج والتي تسمى Breakpoints.

٧ ـ الاختيار Quit

يغلق جميع الملفات المفتوحة وينهي البرنامج ويعود بك إلى محث نظام التشغيل DOS

٢) قائمة *Display*

يوضح شكل ٣ـ٩ هذه القائمة وهي تشتمل على أربعة اختيارات هي :

١ ـ الاختيار Expression

بمجرد اختيار هذا الاختيار يفتح لك مكتشف الأخطاء نافذة لتكتب فيها تعبيرا ما ليتم تجربته عندما يتوقف البرنامج أثناء التنفيذ . ويجب الانتباه إلى أن التعبير الذي تدخله إذا كان صحيحا ولكنه يشير إلى متغير غير موجود بالبرنامج فستحصل على عبارة <undefined> بدلا من رسالة الخطأ (انظر شكل ٤ـ٩) . أما إذا كان يشير إلى متغير موجود بالبرنامج فستحصل على نتيجة منطقية .T. أو .F. (انظر شكل ٥ـ٩) . وللعودة إلى القائمة المنسدلة اضغط مفتاح Esc

٢ ـ الاختيار Trace

يظهر مستطيل يشتمل على معلومات عن البرنامج أو الاجراء الذي يستدعي برنامجا آخرا ويبدأ بالبرنامج الموجود في أعلى مستوى داخل النظام ويظهر رقم السطر

```
Control  Display  Variable  Help  Break  Watch

Expression      Edit        Delete      Retu   Proc SACUST
Trace                                          Line      19
Status                                         Break points <off>
Database

                    Customer main data

Account No.   :
Customer name :
Company Name  :
Address       :
City          :
Phone         :
```

شكل ٣-٩

```
Control  Display  Variable  Help  Break  Watch

Expression      Edit        Delete      Retu   Proc SACUST
Trace                                          Line      19
Status                                         Break points <off>

Expression


Value
  <Undefined>

Company Name  :
Address       :
City          :
Phone         :
```

شكل ٤-٩

```
Control  Display  Variable  Help  Break  Watch

   Expression   Edit      Delete     Retu  Proc SACUST
   Trace                                   Line    19
   Status                                  Break points <off>

 Expression


 Value
   .T.

   Company Name :
   Address      :
   City         :
   Phone        :
```

شكل ٥ـ٩

الذي استدعى البرنامج التالي (انظر شكل ٦ـ٩) وللرجوع إلى القائمة اضغط مفتاح Esc

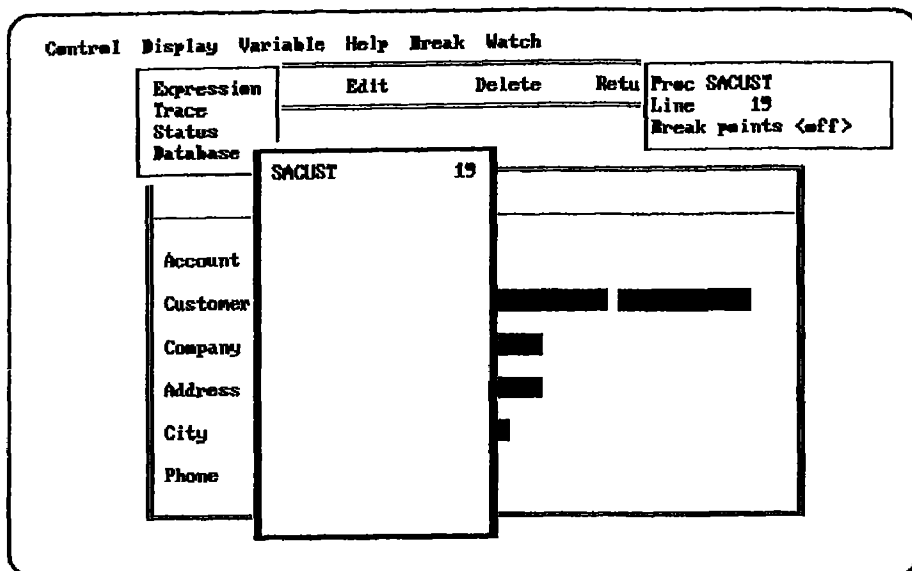
٣ ـ الاختيار Status

يشبه هذا الاختيار أمر SET الموجود في dBASE III PLUS فيظهر حالة الدوال المستخدم في البرنامج (انظر شكل ٧ـ٩). اضغط مفتاح Esc للرجوع إلى القائمة.

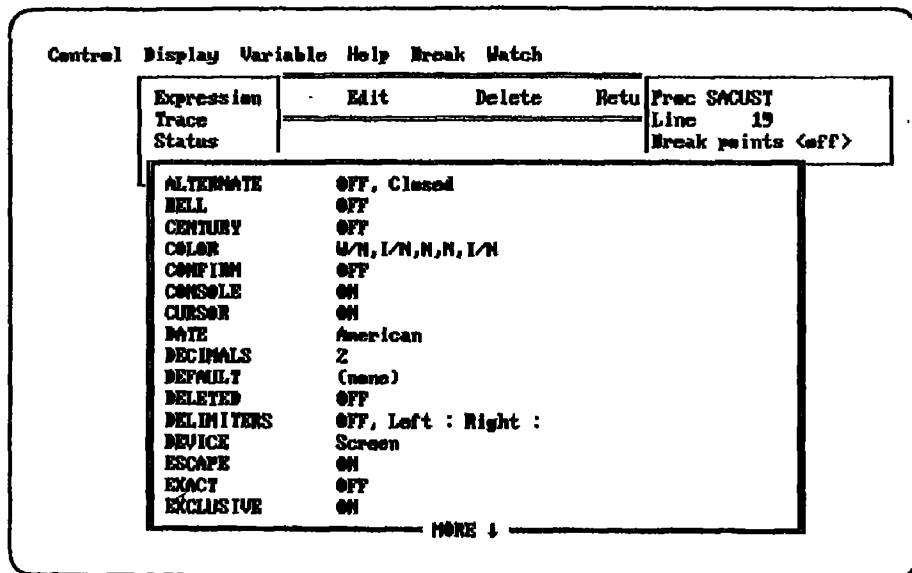
٤ ـ الاختيار Database

يظهر نافذة تتضمن أسماء الملفات المفتوحة. فإذا لم تكن هناك ملفات مفتوحة فتظهر رسالة WAITING في أسفل الشاشة. ويظهر أيضا تبعا لهذا الاختيار وظائف المفاتيح من F1 إلى F4 على النحو التالي:

* مفتاح F1 (overview) لاظهار ملعومات عن الملف المفتوح مثل ملف الفهرسة المختار وعدد السجلات... الخ.
* مفتاح F2 (Relations) العلاقة بين الملفات التي يستخدمها النظام.
* مفتاح F3 (Indexes) ترتيب ملفات الفهرسة والمفتاح المختار لكل فهرس.
* مفتاح F4 (Structure) يظهر اسم ومواصفات ملفات قاعدة البيانات.



شكل ٦ـ٩



شكل ٧ـ٩

٣) قائمة *Variable*

ويتبع هذه القائمة قائمة منسدلة تشتمل على الاختيارات الأربعة التالية (انظر شكل ٨ـ٩).

١ ـ الاختيار Assign Private

يستخدم لانشاء حقول ذاكرة خاصة (Private Memory Variables) أو تعديل تلك الموجودة في البرنامج. وبمجرد اختيار هذا الاختيار فإن مكتشف الأخطاء يطلب منك إدخال اسم المتغير والقيمة الجديدة التي ستخصص له وللعودة إلى القائمة المنسدلة اضغط مفتاح Esc

٢ ـ الاختيار Assign Public

مشابه تماما للاختيار السابق والفرق الوحيد أن هذا الاختيار يتعامل مع حقول الذاكرة التي تنشأ بأمر Public

```
Control  Display  Variable  Help  Break  Watch
                  Assign Private   Delete    Retu  Proc SACUST
                  Assign Public                    Line     19
                  View Privates                    Break points <off>
                  View Publics
                  Customer main data

        Account No.  :
        Customer name:
        Company Name :
        Address      :
        City         :
        Phone        :
```

شكل ٨ـ٩

٣ ـ الاختيار View Private

يظهر نوعين من المعلومات كل نوع داخل مستطيل خاص وكلا المستطيلين يظهر على شاشة واحدة. الأول: يظهر أسماء البرامج الموجودة بالذاكرة والثاني: يظهر حقول الذاكرة التي تنشأ بأمر Private (أسماؤها ـ أنواعها ـ ومحتوياتها). وللعودة إلى القائمة المنسدلة اضغط مفتاح Esc

٤ ـ الاختيار View Public

يظهر معلومات عن حقول الذاكرة التي تنشأ باستخدام أمر Public مثل أسمائها ـ أنواعها ومحتوياتها فإذا لم تكن هناك حقول ذاكرة من هذا النوع فستظهر كلمة WAITING في أسفل الشاشة.

٤) قائمة *Help*

هذه القائمة تظهر معلومات مختصرة عن الاختيارات الست الموجودة بالقائمة الأفقية لمكتشف الأخطاء والاختيارات الموجودة بالقوائم المنسدلة عنها. وقد تناولنا بالشرح حتى الآن الاختيارات الأربعة الأولى. وسنشرح فيما يلي باقي الاختيارات. (انظر شكل ٩ـ٩). وللحصول على معلومات عن أحد الاختيارات المعروضة ضع الشريط المضاء فوقه ثم اضغط مفتاح الادخال. فمثلا لتحصل على معلومات إضافية عن الاختيار Speed key. ضع الشريط المضاء فوقه ثم اضغط مفتاح الادخال. ستحصل على شكل ١٠ـ٩. وهو عبارة عن نافذة تشتمل على وظائف بعض المفاتيح وكلها تستلزم الضغط على مفتاح Alt بالاضافة إلى المفتاح الآخر.

٥) قائمة *Break*

تستطيع تحديد أماكن داخل البرنامج عندما يصل إليها التنفيذ يتوقف البرنامج ويستدعي مكتشف الأخطاء هذه الأماكن تسمى Breakpoints أو نقاط التوقف والحد الأقصى الذي يمكن تحديده داخل البرنامج لهذه النقاط هو ١٦. وأثناء توقف البرنامج في إحدى هذه النقاط يمكنك اختيار المتغيرات والتعبيرات التي يشتمل عليها البرنامج بالاضافة إلى أمور أخرى تمكنك من معرفة حالة البرنامج. وتتيح قائمة Break التعامل مع أربعة اختيارات هي:

```
Control  Display  Variable  Help  Break  Watch

        Add      Control       lete     Retu  Proc SACUST
                 Display                      Line     19
                 Variable                     Break points <off>
                 Help
                 Break
            Cus  Watch
                 Speed Keys
                 About
  Account No.   :

  Customer name:

  Company Name  :

  Address       :

  City          :

  Phone         :
```

شكل ٩-٩

```
Control  Display  Variable  Help  Break  Watch

        Add      Control       lete     Retu  Proc SACUST
                 Display                      Line     19
                 Variable                     Break points <off>

                          Speed keys

               ALT-Z    -    Control menu
               ALT-X    -    Display menu.
               ALT-V    -    Variable menu.

               ALT-W    -    Watch menu.
               ALT-P    -    Set a Watch point and return.

               ALT-H    -    Help menu.
               ALT-F    -    Execute specific Help and return.

               ALT-B    -    Set Break point and return.

               ALT-S    -    Single step and return.
                             MORE ↓
```

شكل ٩-١٠

١ ـ الاختيار Toggle

يسمح بتنشيط نقاط التوقف التي سبق تحديدها داخل البرنامج أو بإبطال مفعولها (انظر شكل ١١ـ٩) . فإذا كان في حالة نشطة (active) واخترت toggle فتصير خاملة (inactive) والعكس صحيح .

٢ ـ الاختيار Line

يخصص نقطة توقف (Breakpoint) اعتمادا على اسم البرنامج أو رقم السطر وفي هذه الحالة يستحثك مكتشف الأخطاء لادخال اسم البرنامج ورقم السطر.

٣ ـ الاختيار Expression

يُعرِّف تعبيرا منطقيا ليعمل كنقطة توقف فإذا وقع هذا التعبير الذي تم تعريفه صحيحا أثناء تنفيذ البرنامج يتوقف التنفيذ ويتم استدعاء مكتشف الأخطاء.

٤ ـ الاختيار Delete

لالغاء نقطة/نقاط توقف سبق تحديدها.

```
Control  Display  Variable  Help  Break  Watch
          Add     Edit     Delete    Retu | Proc SACUST
                                          | Line      19
                                          | Break points <off>
  Toggle  Line  Expression  Delete
     Account No.   :
     Customer name:
     Company Name  :
     Address       :
     City          :
     Phone         :
```

شكل ١١ـ٩

٦) قائمة *Watch*

تتحكم هذه القائمة بصفة عامة في اختيارات القائمة التي تظهر على يمين الشاشة أثناء التعامل مع مكتشف الأخطاء وتسمى Watch box (انظر شكل ١٢-٩). ويتم التحكم في قائمة Watch Box التي تظهر تلقائيا على يمين الشاشة عن طريق قائمة Watch التالية:

١ - الاختيار Toggle

يسمح بالتبديل بين إظهار قائمة Watch Box من عدمه. فإذا كانت ظاهرة واخترت Toggle فستختفي وإذا كانت مختفية واخترت Toggle فستظهر.

٢ - الاختيار Set

يتيح هذا الاختيار ٣ اختيارات إضافية أخرى هي (انظر شكل ١٣-٩).

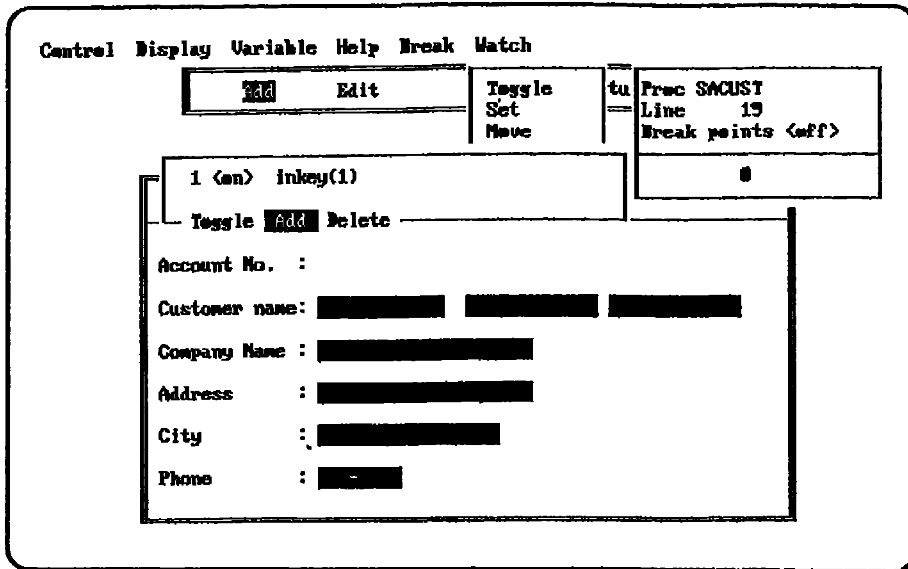
* Toggle: للتبديل بين إظهار أو عدم إظهار قيمة تظهر في Watch Box تسمى Watch points

```
Control  Display  Variable  Help  Break  Watch
          Add      Edit        Toggle     tu  Proc  SACUST
                               Set            Line      19
                               Move           Break points <off>

                  Customer main data

        Account No.   :
        Customer name :
        Company Name  :
        Address       :
        City          :
        Phone         :
```

شكل ١٢ - ٩



شكل ١٣ ـ ٩

* Add: تضيف تعبيرا (Watch point) يؤخذ في الحسبان عند تنفيذ البرنامج .
* Delete: لحذف تعبير (Watch point) تحت إضافته بالاختيار Add.

٣ ـ الاختيار Move

لتغيير مكان قائمة Watch Box من يمين الشاشة إلى يسارها أو العكس .

والمثال التالي يوضح كيفية التعامل مع هذه القائمة .

قلنا إن الاختيار (Go (animation الموجود ضمن قائمة Control يتسبب في إظهار معلومات عن البرنامج الذي يجري تنفيذه داخل مستطيل يسمى Watch Box إلا أن المعلومات تظهر بسرعة عالية جدا بحيث لا يمكنك متابعتها . فإذا أردت أن تخصص ثانية واحدة لكل أمر أثناء التنفيذ لتتمكن من متابعة الأوامر التي يجري تنفيذها لمعرفة آخر أمر يتم تنفيذه قبل وقوع خطأ ما اتبع الخطوات التالية :

اختر قائمة Control ثم (Go (animation

اختر قائمة Watch ثم Set ثم Add ثم اكتب التعبير الآتي

INKEY (1)

(انظر شكل ١٣-٩) ثم تابع تنفيذ البرنامج .

إذا أردت إلغاء هذا التعبير والعودة لتنفيذ البرنامج بالسرعة العادية اختر Delete ثم رد على الرسالة بكتابة الرقم الذي خصص للتعبير المراد إلغاؤه (وهو في هذا المثال الرقم 1).

تــذكــر...!

- لكي تتمكن من استخدام مكتشف الأخطاء (Clipper Debugger) يجب ربط ملف DEBUG.OBJ مع النظام أو البرنامج المقترح ويجب أن تكتب اسم ملف DEBUG بعـد اسم الـبرنامج الذي تنوي تجربته باستخدام مكتشف الأخطاء. وإلا فلن تستطيع تشغيل برنامجك بطريقة صحيحة.
- يتم استدعاء مكتشف الأخطاء بإحدى ثلاث طرق: ضغط مفتاح Alt-D أو عندما يصل البرنامج أثناء التنفيذ إلى الوظيفة AltD() أو عندما يكتشف البرنامج إحدى نقاط التوقف (Break points).
- إذا انتهيت من تجربة النظام وتنقية أخطائه أعد ربطه مرة أخرى بدون إضافة ملف DEBUG.OBJ لتوفر المساحة التي يشغلها داخل برنامجك.
- تشتمل حزمة «كلبر» على ملف آخر اسمه ALTERROR.PRG يساعد أيضا في اكتشاف بعض الأخطاء ولذلك يمكنك ترجمته وربطه مع ملف DEBUG.OBJ للاستفادة منه.

الفصل العاشر

التعامل مع أخطاء البرنامج

يشرح هذا الفصل الوظائف الت الموجودة بملف ERRORSYS.PRG والتي تتعامل مع أخطاء البرامج. ويوضح أيضا كيف يمكنك تعديل محتويات أي منها باستخدام أمر BEGIN SEQUENCE لتناسب حاجتك الخاصة. أو تعديلها لتوجيه رسائل الخطأ إلى ملف نصي.

ويشرح بعد ذلك الأخطاء التي لا دخل للمبرمج فيها غالبا والرسائل التي تنتج تبعا لها.

كيف يمكن تقليل احتمالات الخطأ

من الأفضل مراجعة البرنامج في صورته المصدرية قبل ترجمته بواسطة «كلبر». لأن معظم أخطاء البرنامج تكون نتيجة أخطاء إملائية أو أخطاء في صياغة الأوامر. وهذه يسهل اكتشافها وتصحيحها قبل ترجمة البرنامج. وإذا اكتشف المترجم (Compiler) أخطاء أثناء ترجمة البرنامج فإنه يظهر رسالة تحدد نوع الخطأ ورقم السطر واسم البرنامج الذي كان ينفذ عند حدوث هذا الخطأ. ويتم إلغاء العمل بدون تصحيح هذا الخطأ.

وتتميز «كلبر» بأنها تعطي المبرمج الفرصة للسيطرة على أخطاء البرنامج التي يكتشفها المترجم. باستثناء الأخطاء التي تخرج عن سيطرة المبرمج مثل تلك التي تأتي من برامج خارجية مثل برامج «سي» أو «أسمبلي» أو تلك الناتجة عن مكونات الحاسب مثل سوء توصيل الطابعة بالحاسب نتيجة قطع السلك أو عدم تركيب ورق الطباعة أو امتلاء القرص الممغنط بالبيانات. . . الخ.

والأخطاء التي يمكن تصحيحها أو تجنبها كثيرة وأسبابها أيضا كثيرة. فمثلا قد تخطىء في كتابة اسم الحقل وبالتالي ستحصل على رسالة تفيد أن الحقل غير موجود أو قد تجمع عبارة رقمية مع أخرى حرفية وبالتالي ستحصل على رسالة تفيد أن البيانات غير متطابقة. . . وهكذا. وهناك أخطاء يمكن توقعها وتجنب حدوثها وإخبار البرنامج بالتصرف اللازم حال حدوثها ومن أمثلتها ما هو التصرف إذا حاول المستفيد إدخال رقم موظف موجود من قبل أو إذا حاول تعديل محتويات سجل أو ملف ووجده مغلقا؟ ومن المفيد استخدام أمر BEGIN SEQUENCE لتجنب حدوث خطأ أو أخطاء تتسبب في إنهاء البرنامج والرجوع إلى محث نظام التشغيل

ملاحظة: راجع أمر BEGIN SEQUENCE بالتفصيل في الفصل الثاني عشر من الباب الثالث.

وأمر BEGIN SEQUENCE يشبه في تركيبه أمر DOWHILE ويأخذ هذا الشكل

```
BEGIN SEQUENCE
            <Commands>
BREAK
            <Commands>
END
            <Commands>        && Commands to avoid error
```

وعنـدمـا تكتشف «كلبر» أمر BREAK تنقل تنفيذ البرنامج إلى الأمر التالي لعبارة END. وللتوضيح نسوق المثال التالي.

ما هو الحل إذا كانت الطابعة غير جاهزة وأراد البرنامج أن يطبع أحد التقارير؟ إذا كانت الطابعة غير جاهزة لأي سبب فإن البرنامج سيظهر رسالة تفيد أن الطابعة غر جاهزة أو غير موصلة وسينهي التنفيذ ويعود إلى نظام التشغيل. وتجنبا لحدوث هذا الأمر يمكن استخدام أمر BEGIN SEQUENCE لتوجيه البرنامج لتنفيذ أمر/أوامر معينة في حالة فشل الطابعة على النحو التالي:

```
BEGIN SEQUENCE
  DO WHILE .T.
    IF ISPRINTER()
      REPORT FORM STKRPRT TO PRINT
    ELSE
      BREAK
    ENDIF
  ENDDO
END
* < الأوامر المطلوب تنفيذها إذا كانت الطابعة غير جاهزة >
```

وفي هذا الجزء من البرنامج يتم طباعة التقرير إذا كانت الطابعة جاهزة فقط وإلا تنفذ الأوامر التي تلي أمر END وقد توجه هذه الأوامر البرنامج للعودة للبرنامج الرئيسي مرة أخرى بدلا من إنهائه أو إلغائه.

أنواع الأخطاء التي يمكن تصحيحها

تقسم الأخطاء التي يمكن الحصول عليها من البرامج وتصحيحها إلى الأنواع الستة التالية :

- أخطاء ملفات قاعدة البيانات (Database Errors)
- أخطاء تعبيرات (Expression Errors)
- أخطاء فتح الملفات (Open Errors)
- أخطاء متنوعة (Miscellaneous Errors)
- أخطا الطباعة (Print Errors)
- أخطاء متغيرات غير معرّفة (Undefined Errors)

ويتم معالجة هذه الأخطاء بواسطة برامج (وظائف خاصة) تأتي مع حزمة «كلبر». وهذه البرامج مكتوبة بشفرة ASCII أي في صورة المصدر وتجدها مجتمعة في ملف ERRORSYS.PRG ويمكن التأكد من وجود هذا الملف بإصدار الأمر التالي تحت محث نظام التشغيل

```
C:\CLIPPER> DIR ERRORSYS.PRG
```

فإذا تأكدت من وجوده فيمكنك طباعته أو تعديله بأي منسق للنصوص تستخدمه. والهدف من وضع هذه البرامج (أو الوظائف الخاصة) في صورتها المصدرية هو إعطاؤك الفرصة لتعديل هذه البرامج لتناسب حاجتك الخاصة. فإذا رغبت في ذلك وكنت قادرا على تعديل هذه البرامج، (سوف نشرح هذه البرامج بعد قليل). فيجب أن تضعها كلها داخل ملف ثم تترجمه إلى ملف OBJ. ثم اربط هذا الملف مع النظام أو الرنامج الذي تستخدمه ولا تنسى أن تكتب اسم برنامجك الرئيسي أولا بعد عبارة FILES (راجع الفصل السادس) ولأنك ستستخدم هذه الوظائف بنفس الاسم المخصص لها في مكتبة «كلبر» CLIPPER.LIB فإن الوظائف التي عُدلت هي التي سيتم ربطها بدلا من الوظائف الأساسية الموجودة بنفس الاسم والتي تخصصها «كلبر».

ويوضح الجدول التالي أنواع الأخطاء وأسماء البرامج (الوظائف الخاصة) التي تعالج كلا منها.

نوع الخطأ	الوظيفة المخصصة له
أخطاء ملفات قاعدة البيانات (Database Errors)	DB_ERROR()
أخطاء تعبيرات (Expression Errors)	EXPR_ERROR()
أخطاء متنوعة (Miscellaneous Errors)	MISC_ERROR()
أخطاء فتح الملفات (Open Errors)	OPEN_ERROR()
أخطا الطباعة (Print Errors)	PRINT_ERROR()
أخطاء متغيرات غير موجودة (Undefined Errors)	UNDEF_ERROR()

وسوف نشرح هذه الوظائف الخاصة بعد قليل ليمكنك بعد ذلك استخدامها كما هي أو تعديلها حسب حاجتك لاحكام الرقابة على الأخطاء التي تحصل عليها من برامجك أو تسجيل رسائل الخطأ على ملف مستقل.

ملاحظة: يشتمل ملف ALTERROR.PRG على مجموعة من الوظائف الجاهزة التي يستخدمها مكتشف الأخطاء (Debugger) وبعضها يشبه الوظائف التي بين أيدينا.

ويتم قبول المعطيات التالية لكل الوظائف السابقة والتي تتعامل مع أخطاء البرنامج (بعضها يمكن أن يقبل أكثر وسنوضحها في حينها).

- Name (الاسم) اسم البرنامج أو الاجراء الذي كان ينفذ ساعة حدوث الخطأ.
- Line (السطر): رقم السطر الذي حدث عنده الخطأ.
- Info (المعلومات): رسالة تشرح الخطأ.

وفيما يلي سنشرح أنـواع الأخـ سابقة والوظائف الموجودة بمكتبة «كلبر» لمعالجة هذه الأخطاء. وسنبين على المثال كيفية تعديل إحدى هذه الوظائف لتسجيل رسالة الخطأ على ملف نصي مـ ـت

أخطاء ملفات قواعد البيانات Database Errors

المقصود بأخطاء ملفات قواعد البيانات الأخطاء التي تخص الملفات. ولكنها لا تشمل عمليات فتح وغلق الملفات. وعند حدوث أحد هذه الأخطاء يتم استدعاء الوظيفة DB_ERROR ويشتمل شكل ١٠-١ على النص الكامل للوظيفة DB_ERROR كما جاءت بملف ERRORSYS.PRG

دقق النظر في الأوامر التي تشتمل عليها هذه الوظيفة تلاحظ ما يلي:

أولا: أن هذه الوظيفة تتعرف على ٣ معطيات هي:

أ - Name: اسم البرنامج الذي تسبب في الخطأ.

ب - Line: رقم السطر الذي حدث عند الخطأ. وهذا الرقم هو الذي يخصصه المترجم لكل سطر من سطور البرنامج. فإذا كنت استخدمت الاختيار l- (بمعنى لا تعطي أرقاما للسطور) فإن هذا السطر سيحتوي على رقم صفر.

```
***
*       db_error(name, line, info)
*

FUNCTION db_error
PARAM name, line, info

SET DEVICE TO SCREEN
@ 0, 0
@ 0, 0 SAY "Proc " + M->name + " line " + LTRIM(STR(M->line)) +;
                    ", " + M->info

NOTE BREAK

QUIT

RETURN .F.
```

شكل ١٠-١

جـ ـ Info: وصف النوع الخطأ الذي حدث. وهذا الوصف يمكن أن يكون إحدى الرسائل التالية:

■ Database Required

وتنتج هذه الرسالة إذا كان الملف غير مفتوح وأصدر البرنامج أحد الأوامر التي تتطلب ملفا مفتوحا مثل AVERAGE أو APPEND BLANK

■ Lock Required

تنتج هذه الرسالة إذا وجدت «كلبر» أن الملف مفتوح وأصدر البرنامج أحد الأوامر التي تتطلب غلق الملف أو السجل (ويحدث هذا في حالة استخدام شبكة اتصالات محلية).

■ Exclusive Required

تنتج هذه الرسالة إذا وجدت «كلبر» أن الملف يستخدم استخداما جماعيا (Shared) في حين يتطلب البرنامج أن يستخدم الملف استخداما منفردا (Exclusive) ويحدث هذا الخطأ في حالة استخدام شبكة اتصالات محلية.

■ Field Numeric Overflow

تحصل هذه الرسالة إذا وجدت «كلبر» أن عدد خانات الحقل أقل من الرقم المطلوب وضعه فيها.

■ Index File Corrupted

تحصل هذه الرسالة عندما تكتشف «كلبر» أن ملف الفهرس أصبح تالفا.

ثانيا أمر SET DEVICE TO SCREEN يتأكد أن الرسالة ستظهر على الشاشة لأنه ربما أن البرنامج يستخدم الطابعة في ذلك الوقت.

ثالثا أمر 0,0@ يحذف البيانات الموجودة في السطر الأول من الشاشة (سطر رقم صفر).

رابعا أمر ...SAY 0,0@ يظهر المعطيات الثلاثة (اسم البرنامج ورقم السطر ونوع الخطأ) في السطر الأول من الشاشة.

خامسا أمر NOTE BREAK تعليق أو ملاحظة لن تنفذها «كلبر» ووضعه هنا ليذكرك أنك قد تفضل استخدام أمر BREAK في هذا المكان داخل أمر BEGIN SEQUENCE بدلا من أمر QUIT التالي. وبذلك تتفادى إنهاء البرنامج وإعطاء تعليمات أخرى.

ويعتمد قرار «كلبر» على القيمة التي تشتمل عليها هذه الوظيفة .T. أو .F. فطبقا للوضع الطبيعي لهذه الوظيفة فإنها ستشتمل على القيمة المنطقية .F. وبالتالي فإن البرنامج سينتهي.

تعديل الوظيفة DB_ERROR() لتسجيل رسالة الخطأ على ملف نصي

يشتمل شكل ٢-١٠ على التعديل المقترح للوظيفة DB_ERROR() ليتم تسجيل رسالة الخطأ على ملف نصي اسمه ERR.ASC ومن هذا الشكل تلاحظ أننا أظهرنا رسالة للمستفيد في السطر الأول من الشاشة ثم أغلقنا الملفات لأننا سننشىء ملفا جديدا ثم استخدمنا ٣ وظائف لانشاء وكتابة وغلق ملف نصي هي FCREATE() و FWRITE() و FCLOSE() (راجع الفصل الثالث عشر إذا كنت لا تعرف معنى هذه الوظائف).

أخطاء التعبيرات Expression Errors

أخطاء التعبيرات هي الأخطاء التي تحدث نتيجة تركيب خاطىء لأحد التعبيرات مثال ذلك محاولة جمع محتويات حقلي رقمي مع آخر حرفي في نفس التعبير.

انظر المثال التالي:

```
AMNT="Net Salary"+10000
```

هذا المثال يتسبب في حدوث خطأ لأن البيانات التي يشتمل عليها التعبير غير متطابقة. وعند حدوث أحد هذه الأخطاء يتم استدعاء الوظيفة EXPR_ERROR()

```
***
* FUNCTION DB_ERROR
* DB_ERROR تشبه الوظيفة
* مع فارق واحد هو انها تسجل رسالة الخطأ
* إلى ملف نصي اسمه ERR.ASC قبل انهاء البرنامج
*
FUNCTION DB_ERROR
  PARAM NAME, LINE, INFO
  SET DEVICE TO SCREEN
  @ 0, 0
  @ 0, 0 SAY "Error recording now"
  CLOSE ALL
  F_ERR = FCREATE("ERR.ASC")
  AA= "Proc"+ M->NAME+"Line"+ LTRIM(STR(M->LINE))+","+M->INFO
  FWRITE(F_ERR,AA,LEN(AA))
  FCLOSE(F_ERR)
  QUIT
RETURN .F.
```

شكل ٢ـ١٠

ويشتمل شكل ٣ـ١٠ على النص الكامل للوظيفة EXPR_ERROR() كما جاءت بملف ERRORSYS.PRG

دقق النظر في الأوامر التي تشتمل عليها هذه الوظيفة تلاحظ أن هذه الوظيفة تتعرف على ٤ معطيات جديدة بالاضافة إلى المعطيات الثلاثة التي شرحناها في الوظيفة السابقة DB_ERROR() وهي name (الاسم)، line (رقم السطر)، info (رسالة الخطأ) وهذه المعطيات الجديدة هي:

model_1_2_3

وسوف نعود لشرح هذه المعطيات بعد أن نوضح الرسائل التي يمكن أن يشتمل عليها الجزء info

قلنا إن الجزء info يشتمل على وصف لنوع الخطأ الذي حدث وهذا الوصف يمكن أن يكون إحدى الرسائل التالية:

```
***
*       expr_error(name, line, info, model, _1, _2, _3)
*

FUNCTION expr_error
PARAM name, line, info, model, _1, _2, _3

SET DEVICE TO SCREEN
@ 0, 0
@ 0, 0 SAY "Proc " + M->name + " line " + LTRIM(STR(M->line)) +;
                        ", " + M->info

QUIT

RETURN .F.
```

شكل ٣ـ١٠

● **Type Mismatch**

تنتج هذه الرسالة إذا حاولت ربط عبارتين مختلفتين في تعبير واحد فمثلا العبارة التالية تربط عبارة حرفية مع أخرى رقمية

AA= "ok" + 123

● **Subscript range**

تنتج إذا أردت استخدام عنصر من عناصر مصفوفة غير موجود.

● **Zero divid**

تنتج إذا حاولت قسمة رقم على صفر. وطبعا هذا غير صحيح.

وإليك شرح المعطيات الأربعة الجديدة في هذه الوظيفة.

■ **Model** يشتمل هذا الجزء على التعبير الذي تسبب في الخطأ. وهذا التعبير يوضع في صورة تعبير حرفي (expC).

■ 1ـ أول جزء من التعبير الخاطىء.

■ 2ـ ثاني جزء من التعبير الخاطىء.

■ 3ـ ثالث جزء من التعبير الخاطىء.

وكما نلاحظ أن كلا من 1ـ و 2ـ و 3ـ ستشتمل على القيم الموجودة بالتعبير ساعة حدوث الخطأ. وللتوضيح نسوق المثال التالي بفرض أن التعبير الخاطىء كان كما يلي:

AA= "ok" + 123

فإن المتغير 1ـ سيشتمل على القيمة "ok"
والمتغير 2ـ سيشتمل على القيمة 123
أما الجزء info فسيشتمل على الرسالة التالية في هذه الحالة

Type mismatch

أما باقي أوامر الوظيفة فهي مشابهة لأوامر الوظيفة السابقة DB_ERROR() وتقوم أيضا بوظيفة إظهار رسالة وإنهاء البرنامج (راجع الشرح السابق). ويمكن أيضا تعديل محتويات هذه الوظيفة لتسجيل رسالة الخطأ على ملف نصي بنفس الطريقة التي اتبعناها في البرنامج الموجود في شكل ٢-١٠ السابق.

أخطاء متنوعة Miscellaneous Errors

هذه الأخطاء هي الأخطاء التي لم ترد في المجموعات الخمس الأخرى وهي عادة أخطاء بسيطة.

ويشتمل شكل ٤-١٠ على الوظيفة MISC_ERROR() التي تتعامل مع هذه الأخطاء.

وفي ضوء الشرح الذي تقدم للوظيفتين السابقتين تستطيع تَفهُّم محتويات هذه الوظيفة والغرض منها.

```
***
*       misc_error(name, line, info, model)
*

FUNCTION misc_error
PARAM name, line, info, model

SET DEVICE TO SCREEN
@ 0, 0
@ 0, 0 SAY "Proc " + M->name + " line " + LTRIM(STR(M->line)) +;
                    ", " + M->info

NOTE BREAK

QUIT


RETURN .F.
```

شكل ٤-١٠

ويعتمد قرار «كلبر» على القيمة التي تشتمل عليها هذه الوظيفة (.T. أو .F.) فإذا كانت .T. فإن «كلبر» ستعيد المحاولة. أما إذاكانت .F. كما هو الوضع الأصلي للوظيفة فإن «كلبر» ستظهر الرسالة وتنهي البرنامج. والجزء model في هذه الوظيفة يشتمل على الأمر الذي تسبب في حدوث الخطأ وأيضا يمكنك تعديل أوامر هذه الوظيفة لتكتب رسالة الخطأ على ملف ببرنامج مشابه للموجود في شكل ٢-١٠.

ونوضح فيما يلي رسائل الخطأ التي يمكن أن يشتمل عليها الجزء info من الوظيفة.

● **Type Mismatch**

تنتج هذه الرسالة إذا حاول البرنامج استخدام أمر REPLACE لاستبدال محتويات حقل بنوع من البيانات لا يوافق النوع الذي يشتمل عليه الحقل مثل استبدال حقل رقمي بالعبارة الحرفية "100".

ملاحظة: الفرق بين هذه الرسالة ورسالة ***Type Mismatch*** *التي تنتج من الوظيفة* ***EXPR_ERROR()*** *أن هذه الرسالة نتيجة لأمر يغير في محتويات الملف أما رسالة الوظيفة* ***EXPR_ERROR()*** *فهي نتيجة لمحتويات تعبير خطأ ولا يسبب تغيرا في محتويات الملف.*

- **Run Error**

تنتج هذه الرسالة إذا حاولت تشغيل أحد البرامج الخارجية التي تتطلب تحميل ملف COMMAND.COM باستخدام أمر RUN

د أخطاء فتح الملفات Open Errors

المقصود بأخطاء فتح الملفات الأخطاء التي تحدث عندما نريد فتح أحد الملفات. وتحدث معظم هذه الأخطاء عند استخدام شبكة اتصالات محلية (LAN) نتيجة محاولة فتح الملف لاستخدامه استخداما منفردا في الوقت الذي يستخدم فيه نفس الملف من قبل مستفيد آخر (راجع الفصل الحادي عشر استخدام «كلبر» مع شبكة الاتصالات).

ويشتمل شكل ٥ـ١٠ على الوظيفة OPEN_ERROR() التي تتعامل مع أخطاء فتح الملفات. وعن هذا الشكل نوضح ما يلي:

■ تبدأ الوظيفة بأمر IF NETERR() ومعناه إذا استخدم المستفيد أمر USE خطأ فإن الوظيفة NETERR() ستكون صحيحة (.T.) لأن خطأ قد حدث. وبالتالي فإن الوظيفة OPEN_ERROR() ستنتهي.

■ نوضح فيما يلي المعطيات التي تشتمل عليها هذه الوظيفة:
- Name و Line لا تختلف عما سبق شرحه في الوظائف السابقة.
- Info : تشتمل دائما على رسالة واحدة وهي Open error بمعنى أن الملف لم يفتح لسبب ما.
- Model : تشتمل على الأمر الذي تسبب في وقوع الخطأ.
- 1 وتشتمل على اسم الملف الذي لم يفتح.

■ تشتمل هذه الوظيفة على أوامر جديدة بخلاف التي شرحناها في الوظائف السابقة لتظهر رسالة لتعطيك الفرصة لاعادة المحاولة بهذا الشكل

Retry? (Y/N)

فإذا اخترت Y (راجع أمر IF .NOT. داخل الوظيفة) فسيتم إعادة المحاولة أما إذا اخترت الرد N فإن البرنامج الذي تسبب في الخطأ سينتهي نتيجة أمر QUIT

```
***
*       open_error(name, line, info, model, _1)
*

FUNCTION open_error
PARAM name, line, info, model, _1

IF NETERR() .AND. model == "USE"
        RETURN .F.
END

SET DEVICE TO SCREEN
@ 0, 0
@ 0, 0 SAY "Proc " + M->name + " line " + LTRIM(STR(M->line)) +;
", " + M->info + " " + M->_1 + " (" + LTRIM(STR(DOSERROR())) + ")"
@ 0, 65 SAY "Retry? (Y/N)"

INKEY(0)

DO WHILE .NOT. CHR(LASTKEY()) $ "YyNn"
        INKEY(0)
END

IF .NOT. CHR(LASTKEY()) $ "Yy"
        QUIT
END

@ 0,0

RETURN .T.
```

شكل ٥ـ١٠

أخطاء الطباعة Print Errors

إذا حاول البرنامج استخدام طابعة غير جاهزة للطباعة أو غير متصلة بالحاسب فستظهر «كلبر» رسالة خطأ. وتكون الطابعة غير جاهزة للطباعة إذا كانت مطفأة أو كان الورق غير مركب عليها. وتحدث أخطاء الطباعة أيضا عندما يرسل أكثر من مستفيد طباعة إلى نفس الطابعة - في حالة استخدام شبكة اتصالات محلية - وتكون محطة الاستقبال الخاصة بالطابعة (Buffer) ممتلئة جدا. وتحدث كل هذه الأخطاء إذا كانت الطابعة مركبة على التوازي (Parallel) أما إذا كانت مركبة بالتسلسل (Serial) فإن الخطأ الذي يحدث يكون من نوع خطأ فتح الملفات.

ويشتمل شكل ٦-١٠ على النص الكامل للوظيفة ()PRINT_ERROR التي تتعامل مع أخطاء الطباعة.

وعن هذا الشكل نوضح ما يلي:

- المعطيات التي تستخدمها الوظيفة هي Name لاسم البرنامج، و Line لرقم السطر الذي حدث عنده الخطأ.
- الرسالة الوحيدة التي تظهر نتيجة هذا الخطأ هي:

Printer error

- تشتمل الوظيفة على أوامر لاظهار رسالة لاعادة المحاولة بهذا الشكل:

Retry? (Y/N)

وهي مشابهة للأوامر الموجودة في الوظيفة السابقة ()OPEN_ERROR

- إذا اختار المشغل الرد N لعدم تكرار المحاولة فإن «كلبر» ستتجاهل الأمر الذي تسبب في الخطأ. ولذلك فقد تحصل على العديد من رسائل الخطأ إذا كان البرنامج يستخدم بعد ذلك أكثر من أمر يتطلب استخدام الطابعة.

أخطاء متغيرات غير موجودة Undefined Errors

هذا النوع من الأخطاء يرجع إلى استخدام حقل أو متغير غير معرَّف. وهو ما يحدث إذا استخدمت الوظيفة ()Type للسؤال عن نوع البيانات المخزنة داخل حقل

```
***
*       print_error(name, line)
*

FUNCTION print_error


PARAM name, line

SET DEVICE TO SCREEN
@ 0, 0
@ 0, 0 SAY "Proc " + M->name + " line " + LTRIM(STR(M->line)) +;
                        ", printer not ready"

@ 0, 65 SAY "Retry? (Y/N)"

INKEY(0)

DO WHILE .NOT. CHR(LASTKEY()) $ "YyNn"
       INKEY(0)
END

IF .NOT. CHR(LASTKEY()) $ "Yy"
       QUIT
END

@ 0,0

RETURN .T.
```

شكل ٦-١٠

أو متغير. فإذا كان الحقل غير موجود أو غير معرّف فستحصل على الرد U بمعنى Undefined.

وقد تكون المتغيرات أو الحقول الغير موجودة اسم حقل داخل الملف أو حقل بالذاكرة أو اسم إجراء أو وظيفة مطلوب تنفيذها.

يشتمـل شكـل ٧ـ١٠ على الـوظيفة UNDEF_ERROR() التي تتعامل مع أخطاء المتغيرات الغير موجودة. وإليك شرح المعطيات التي تشتمل عليها هذه الوظيفة.

```
***
*       undef_error(name, line, info, model, _1)
*

FUNCTION undef_error
PARAM name, line, info, model, _1

SET DEVICE TO SCREEN
@ 0, 0
@ 0, 0 SAY "Proc " + M->name + " line " + LTRIM(STR(M->line)) +;
                     ", " + M->info + " " + M->_1

QUIT

RETURN .T.
```

شكل ٧ـ١٠

■ Name لاسم البرنامج ، و Line لرقم السطر الذي حدث عنده الخطأ.

■ Info : تشتمل على إحدى الرسائل التالية:

● **Undefined identifier**

تنتج هذه الرسالة إذا حاول البرنامج استخدام حقل ذاكرة (Memory variable) غير موجود.

● **Not an array**

تنتج هذه الرسالة. إذا حاول البرنامج استخدام عنصر داخل مصفوفة وكان هذا العنصر غير موجود أو كانت المصفوفة كلها غير موجودة.

● **Missing external**

تنتج هذه الرسالة إذا حاول البرنامج استخدام وظيفة من وظائف «كلبر» غير موجودة بالمكتبة EXTEND.LIB

■ Model : تشتمل على الأمر الذي تسبب في وقوع الخطأ.

■ 1ـ : اسم المتغير أو الحقل الذي لم تجده «كلبر».

الأخطاء التي لا يمكن تصحيحها

بالاضافة إلى الأنواع الستة التي شرحناها بالتفصيل فيما سبق قد نحصل على رسائل خطأ لا ترجع إلى خطأ في البرنامج نفسه وإنما ترجع لأسباب غالبا لا دخل للمبرمج فيها ونوضح فيما يلي الرسائل التي يمكن الحصول عليها نتيجة وقوع أحد هذه الأخطاء.

Internal error

سبب هذه الرسالة أن تلفا حدث في ملف الفهرس وعادة تظهر الرسالة ويتوقف البرنامج حتى يتم ضغط أحد المفاتيح فينتهي البرنامج ويرجع إلى محث نظام التشغيل.

Disk full

ترجع هذه الرسالة إلى أن البرنامج يريد أن يكتب ملفا جديدا ولكن المساحة المتبقية على القرص المستخدم غير كافية. وعادة يتوقف البرنامج بعدها ليعطي المشغل الفرصة في إعادة المحاولة.

فإذا اختار المشغل الرد لا (No) فسينتهي البرنامج ويرجع إلى محث نظام التشغيل.

Multiple error

ترجع هذه الرسالة إلى أخطاء شديدة تسببت في خطأ في إحدى الوظائف الست التي تعالج الأخطاء والتي شرحناها قبل قليل ولذلك فإن الخطأ يصير مركبا. وعادة يتوقف البرنامج بعد هذه الرسالة حتى يتم ضغط أحد المفاتيح فينتهي ويرجع التنفيذ إلى محث نظام التشغيل.

Out of memory

ترجع هذه الرسالة إلى أن البرنامج وصل إلى آخر جزء في الذاكرة RAM يمكن استخدامه وعادة يتوقف البرنامج بعد هذه الرسالة حتى يتم ضغط أحد المفاتيح

فينتهي ويرجع التنفيذ إلى محث نظام التشغيل.

Not enough memory

ترجع هذه الرسالة إلى أن «كلبر» استشعرت أن النظام والبرنامج المطلوب تنفيذه لن تستوعبه الذاكرة لكبر حجمه وعادة يتوقف البرنامج بعد هذه الرسالة حتى يتم ضغط أحد المفاتيح فينتهي ويرجع التنفيذ إلى محث نظام التشغيل.

رسائل الأخطاء

بالاضافة إلى الأخطاء التي شرحناها حتى هذه اللحظة قد تحصل على أخطاء أخرى أثناء تجربة البرنامج أو النظام وهذه الأخطاء تنتج إما في مرحلة الترجمة (compiling) أو مرحلة الربط (linking).

ونظرا لكثرة هذه الأخطاء واختلاف أسبابها وبالتالي كثرة الرسائل التي تنتج تبعا لها. فلست في حاجة لقراءة هذه الرسائل بالتتابع مثل قراءة أحد الفصول التي تقدمت والتي تتولى شرح موضوع معين من أوله إلى آخره. وإنما يكفي أن ترجع إلى كتاب الشركة المنتجة إذا حصلت على إحداها لتعرف ماذا تقصد.

ونوضح فيما يلي المقصود بأخطاء مرحلة الترجمة وأخطاء مرحلة الربط.

أخطاء مرحلة الترجمة (Compiling errors)

إذا اكتشفت «كلبر» أخطاء في البرامج المصدرية أثناء ترجمتها فإنها تظهر رسالة لكل خطأ في البرنامج على الشاشة ما لم تختر ملفا أو وحدة أخرى لاظهار رسائل الخطأ عليها غير الشاشة. فيمكنك اختيار إظهار رسائل الخطأ التي تكتشف أثناء الترجمة على ملف معين أو على الطابعة باستخدام علامات إعادة التوجه الموجودة في DOS. فمثلا يمكنك اختيار الطابعة لتظهر عليها رسائل الخطأ باستخدام الأمر التالي:

```
CLIPPER <programname >    PRN
```

ويظهر مع رسالة الخطأ رقم السطر الذي حدث عنده الخطأ وشكل الأمر الخطأ (انظر شكل ٨-١٠).

والأخطاء التي تظهر في مرحلة الترجمة يكون معظمها بسبب خطأ في البرنامج نفسه. ولذلك فيصعب حصر هذه الأخطاء أو الرسائل التي تظهر تبعا لها. وعادة تكون هذه الرسائل واضحة في وصف الخطأ فإذا حصلت على إحدى هذه الرسائل فينصح بمراجعة كتاب الشركة المنتجة للتعرف على وصف الخطأ الذي يخص الرسالة التي حصلت عليها.

```
Compiling SAMAIN.PRG
line 38:  rest of line ignored
@ 0,2  ASY "Date: "+ DTOC(DATE())

1 error
Code Pass 1
Code Pass 2
Code size 787, Symbols 240, Constants 445
```

شكل ٨ـ١٠

أخطاء مرحلة الربط (Linking errors)

إذا أخطأ المبرمج في التعليمات التي توجه برنامج الربط Plink86 plus فسيحصل على رسالة خطأ. تتضمن هذه الرسالة السطر الذي حدث عنده الخطأ متبوعا بعلامات استفهام ??? في مكان الخطأ الذي حدث. (انظر شكل ٩ـ١٠) وإذا ظهرت علامات الاستفهام في بداية السطر فينصح بمراجعة السطر السابق لأنه في الغالب هو الذي سبب الخطأ.

وفي بعض الأحيان تكون هذه الأخطاء بسيطة بحيث لا تؤثر على طريقة تنفيذ البرنامج. وفي هذه الحالة فإن الرسالة التي تظهر تكون رسالة تحذيرية (Warning message) وهي تنبه عن خطأ يمكن تجاهله في السطر المذكور. أما الأخطاء التي تؤثر على تنفيذ البرنامج أو تمنع تنفيذه فهذه لا يمكن تجاهلها. ولذلك فإن «كلبر» تظهر هذه الرسالة وتوقف تنفيذ البرنامج.

ولكل رسالة من رسائل الخطأ أو الرسائل التحذيرية رقم خاص بها. فإذا حصلت على إحدى هذه الرسائل ننصح بمراجعة كتاب الشركة المنتجة للتعرف على وصف الخطأ الذي يخص الرقم الذي حصلت عليه.

```
C:\DBMS\CLIPPER>plink86 fle samain lib clipper,extend
PLINK86plus ( Nantucket ) Version 2.24.
Copyright (C) 1987 by Phoenix Technologies Ltd.,
All Rights Reserved.


fle?? samain lib clipper,extend

Syntax error 11
Expecting statement
C:\DBMS\CLIPPER>
```

شكل ٩ـ١٠

تـذكـر...!

يوضح الجدول التالي باختصار أنواع الأخطاء الست التي يمكن تصحيحها وأسماء الوظائف التي تتعامل معها بالاضافة إلى معطيات كل منها.

الخطأ	الوظيفة	المعطيات
قاعدة البيانات Database	DB_ERROR()	DB_ERROR(name, line, info)
التعبيرات Expression	EXPR_ERROR()	EXPR_ERROR(name, line, info, model,_1,_2,_3)
متنوعة Miscellaneous	MISC_ERROR()	MISC_ERROR(name, line, info, model)
فتح الملفات Open	OPEN_ERROR()	OPEN_ERROR(name, line, info, model, _1)
طباعـة Print	PRINT_ERROR()	PRINT_ERROR(name, line, info)
غير معرّفة Undefined	UNDEF_ERROR()	UNDEF_ERROR(name, line, info, model, _1)

الفصل الحادي عشر

استخدام كلبر مع شبكه اتصالات

يخاطب هذا الفصل المهتمين بتشغيل التطبيقات والبرامج من خلال شبكة اتصالات محلية (LAN). فإذا لم تكن عندك شبكة اتصالات محلية أو إذا لم تكن تنوى كتابة برامج لخدمة أكثر من مستفيد في نفس الوقت. فيمكنك الانتقال إلى الفصل التالي. والعودة لهذا الفصل عندما تعن لك حاجة لتطوير برامج تستخدم شبكة الاتصالات المحلية.

شبكة الاتصالات المحلية هي العبارة المرادفة للعبارة الانجليزية المشهورة Local Area Network وتختصر أحيانا هكذا LAN. وهي تعني باختصار شديد ربط مجموعة من الحاسبات سلكيا مع الوحدة الرئيسية التي يمكن عن طريقها القيام بمعظم العمليات مثل التعامل مع الملفات والطابعات وجميع البرامج والأجهزة المطلوبة.

وقد تزايد الاهتمام بتشبيك الحاسبات في السنوات الأخيرة وذلك لما تحققه من مزايا عديدة مثل المشاركة في استخدام ملحقات الحاسب مثل الطابعات أو البرامج وإمكان اتصال المستخدمين ببعض وإرسال وتلقي الرسائل بينهم. أما مزايا استخدام شبكات الاتصال في نظم إدارة قواعد البيانات فتتلخص في أن أكثر من مستفيد يمكنهم استخدام نفس الملفات في نفس الوقت وباتباع نظام معين للسرية يمكن التحكم في إطلاع أو عدم إطلاع بعض الأشخاص على بعض البيانات الهامة أو السرية.

وتوجد أنواع كثيرة من شبكات الاتصالات التي تستخدم لتشبيك الحاسبات مع بعضها منها على سبيل المثال - ARCent - Etherlink - STARTLAN - Nov el... Dlink الخ وتفضيل واحدة من هذه الشبكات على الأخرى يخرج عن موضوع هذا الكتاب ولذلك فسوف نورد فيما يلي بعض المفاهيم الضرورية عن شبكة الاتصالات المحلية واستخدامها بصفة عامة بصرف النظر عن نوع شبكة الاتصالات التي تستخدمها برامجك.

الحاسب الرئيسي Server

هو الحاسب الرئيسي الذي تتصل به باقي الحاسبات ويمكن للحاسبات الأخرى التعامل مع ملفاته وملحقاته. ويستخدم دائما وحده تخزين كبيرة تسمى File server تقوم بإداء جميع الخدمات للحاسبات الأخرى المتصلة بالحاسب الرئيسي.

الحاسب التابع Client

هو الحاسب الذي يتصل مع الحاسب الرئيسي ويمكنه استخدام ملفاته وملحقاته في حين لا تستطيع باقي الحاسبات التعامل مع ملفاته أو ملحقاته هو. وقد يكون وحدة عرض فقط (Terminal) وفي هذه الحالة يطلق عليه Workstation

وتبعـا لنوع شبكة الاتصالات المستخدمة يمكن استخدام أكثر من حاسب لتقوم بالوظيفة الرئيسية (server) كما يمكن توظيف الحاسب الرئيسي ليقوم بوظيفة التابع (Client).

وفي نظم شبكـات الاتصـالات يخصص اسم وكلمة سر لكل شخص من الأشخاص المسموح لهم بالتعامل مع الشبكة ويتم الدخول إلى الشبكة بعد إعطاء الحاسب الاسم وكلمة السر ويجب أن يكون الاسم وكلمة السر معروفة من قبل. ويحدد لكل مستفيد من الشبكة الملفات والأدلة التي يسمح له بالتعامل معها بالاضافة إلى الخـدمـات الأخـرى مثـل: هل يسمح له بإرسال أعمال إلى الطابعة أو بتغيير محتويات الملفات.

البرمجة لشبكة الاتصالات

يمكن لأكثر من مستفيد في نظم شبكات الاتصالات التعامل مع نفس الملف أو نفس السجـل في نفس الوقت إلا أن هذه الامكانية قد تسبب بعض المشاكل للبيانات ما لم تستخدم بحذر وما لم تتم كتابة البرامج بطريقة تمنع حدوث مثل هذه المشاكل. ويتضح ذلك من المثال التالي:

بفرض أن كلا من أحمد وعبدالله يستخدم حاسبا في مكتبة ويفرض أن كلا الحاسبين مرتبطين بشبكة اتصالات محلية. فإذا كان كلا من أحمد وعبدالله يستخدم ملف الموظفين (EMPLOYEE.DBF) وإذا كان كلاهما يريد تعديل الدرجة المالية للموظف عماد هكذا:

حاسب أحمد	حاسب عبدالله
USE C: EMPLOYEE	USE C: EMPLOYEE
REPLACE DEGREE WITH DEGREE+1	REPLACE DEGREE WITH DEGREE+1

فإذا كان أحمد أسرع من عبدالله وأجرى تعديلاته وأرسلها إلى الملف الرئيسي الموجود على الحاسب الرئيسي (File Server) فإن التعديلات التي أجراها أصبحت

جزءا من الملف وتعـدلت درجـة المـوظف عماد من العـاشرة إلى الحـادية عشرة (١٠+١=١١). أمـا عبدالله الذي تأخر في إجراء تعديلاته فقد أجراها بعد ذلك وأرسلها إلى الحاسب الرئيسي. وبالتالي فإن درجة الموظف عماد بعد تعديل عبدالله أصبحت الثانية عشر (١١+١=١٢) وهذا غير المطلوب.

الاحتمال الثاني أن يكون كلا من أحمد وعبدالله في نفس السرعة وأن تعديلاتهما أرسلت إلى الحـاسب الـرئيسي في نفس الـوقت في هذه الحـالة سيحدث نوع من الارتباك والتصادم يؤدي إلى تلف البيانات وبالتالي لن يمكن استخدام الملفات بعد ذلك.

والحل في مثل هذه الحالات هو إغلاق الملف أو السجل قبل تعديله حتى لا تتـأثـر نفس البيانات بالتعديل الذي يتم من قبل مستفيد آخر في نفس اللحظة. ولذلك فإن مبرمجي نظم إدارة قواعد البيانات الخاصة بشبكة الاتصالات يجب أن يعرفوا جيدا كيف ومتى يفتحون ملفاتهم منفردة أو مشتركة مع آخرين وكيف ومتى يغلقون الملف أو السجل أثناء تعديله أو الاطلاع عليه وأخيرا كيف يحلون مشاكل غلق الملف أو السجل.

وسنـورد فيما يلي الأوامر التي تتحكم في فتح الملف منفردا أو بالاشتراك مع الآخـرين أو التي تتحكم في إغلاق الملف أو السجل قبل تعديله لتجنب مشاكل التعديل التي تتم من قبل أكثر من مستفيد في نفس اللحظة. بالاضافة إلى الأوامر والوظائف الأخرى الضرورية لنظم إدارة قواعد البيانات التي تستخدم شبكة اتصالات.

فتح الملفات

يمكن فتح الملفات لتستخدم بصفة فردية أو بالمشاركة مع الآخرين وتقدم «كلبر» طريقتين لفتح الملفات التي تستخدم مع شبكة الاتصالات استخداما منفردا أو مشتركاً مع آخرين:

الطريقة الأولى: استخدم أمر SET EXCLUSIVE

الـطريقة الثانية: إضافة الاختيار EXCLUSIVE لأمر USE ونوضح فيما يلي كيفية استخدام كلتا الطريقتين.

SET EXCLUSIVE ON/OFF

هذا الأمر من الدوال التي يمكن استخدامها في إحدى حالتين: نعم (ON) أو لا (OFF) والوضع التلقائي لهذا الأمر - ما لم تغيره - هو نعم (ON) وتعني فتح الملف بصفة فردية. بعبارة أخرى منع الآخرين من استخدام نفس الملف في حالة فتحه بواسطة مستفيد بواسطة أمر

USE <filename>

ويسمى هذا الوضع Exclusive Mode ومعناها السماح لمستفيد واحد فقط باستخدام الملف. أما إذا استخدمت الأمر SET EXCLUSIVE OFF فإن «كلبر» ستفهم أنك ستستخدم باقي البرنامج مع شبكة اتصالات محلية وستسمح لأكثر من مستفيد باستخدام نفس الملف في نفس اللحظة.

ويسمى هذا الوضع Shared Mode ومعناها السماح بالمشاركة في استخدام الملف.

USE <filename> EXCLUSIVE

إضافة الاختيار EXCLUSIVE إلى أمر USE معناها منع الآخرين من التعامل مع هذا الملف بكل أوجه التعامل المتاحة مثل قراءته أو تعديله أو الكتابة عليه. بعبارة أخرى إغلاق الملف في وجه الآخرين الموجودين في نفس الشبكة.

المثال الآتي يفتح ملف الموظفين بطريقة مشاركة الاستخدام

```
SELECT 0
USE EMPLOYEE
SET EXCLUSIVE OFF
```

المثال الآتي يفتح ملف الموظفين بطريقة الاستخدام المنفرد

```
SELECT 0
USE EMPLOYEE EXCLUSIVE
```

غلق الملف

تمكن الوظيفة FLOCK() من استخدام الملف استخداما منفردا بصفة مؤقتة. وفي أثناء ذلك يمنع الآخرين من التعديل في الملف بينما يسمح بالاطلاع عليه فقط. وتشتمل هذه الوظيفة على القيمة المنطقية .T. إذا تم إغلاق الملف أما إذا لم تتمكن من إغلاق الملف كما يحدث إذا كان شخص آخر قد أغلقه أو أغلق سجلا موجودا به فإنها تشتمل على القيمة المنطقية .F.

غلق السجل

تمكن الوظيفة RLOCK() من استخدام سجل استخداما منفردا بصفة مؤقتة. وفي أثناء ذلك يمنع الآخرين من التعديل في نفس السجل بينما يسمح لهم بالاطلاع عليه فقط. وتشتمل هذه الوظيفة على القيمة المنطقية .T. أو .F. فإذا تم إغلاق السجل فإنها تشتمل على القيمة .T. أما إذا لم يغلق السجل فإنها تشتمل على القيمة .F. ويحدث ذلك إذا كان شخص آخر قد أغلق السجل أو أغلق الملف الذي يشتمل عليه السجل.

فتح الملفات والسجلات المغلقة

يستخدم أمر UNLOCK لفتح السجل أو الملف الذي أغلق بإحدى الوظيفتين FLOCK() أو RLOCK() والموجود بالمنطقة الحالية. ولا يمكن فتح ملف أو سجل أغلق بواسطة مستفيد آخر.

إذا استخدم هذا الأمر بصيغة:

```
UNLOCK ALL
```

فإن جميع السجلات أو الملفات التي أغلقت بواسطة المستفيد في كل المناطق تعود إلى حالتها السابقة.

الوظيفة *NETERR()*

إذا حاول مستفيد في إحدى المحطات أو الحاسبات المتصلة بالحاسب الرئيسي فتح ملف وكان هذا الملف مستخدما استخداما منفردا بواسطة مستفيد آخر في محطة

أخرى فلن يستطيع ذلك ولن يتم فتح الملف. نفس الشيء يحدث إذا حاول المستفيد فتح الملف بصفة فردية وكان هذا الملف سبق فتحه بواسطة مستفيد آخر بصفة جماعية.

ومن المعروف أن أمر USE لا يعطي رسالة تفيد أن الملف تم فتحه أم لا. لهذا فإن الوظيفة NETERR() تستخدم لتوضح هل تم فتح الملف بنجاح أم لا. وتستخدم الوظيفة NETERR() أيضا لتوضح هل نفذ أمر APPEND BLANK بنجاح أم لا.

وتعطي الوظيفة NETERR() القيمة المنطقية .T. أو .F. والقيمة التلقائية لهذه الوظيفة هي .F. أي Faulse أي لا توجد أخطاء وتستبدل القيمة المنطقية .F. بالقيمة المنطقية .T. إذا حدث خطأ ويحدث خطأ في إحدى الحالات الآتية:

- إذا حاول مستفيد فتح الملف في حين أن مستفيدا آخر في محطة أخرى يستخدم نفس الملف بصفة فردية.
- إذا حاول مستفيد فتح الملف بصفة فردية في حين أن الملف مفتوح بواسطة مستفيد آخر في محطة أخرى.
- إذا حاول مستفيد إضافة سجل جديد بأمر APPEND BLANK في حين أن الملف أغلق مؤقتا من محطة أخرى بالوظيفة FLOCK()
- إذا حاول مستفيد إضافة سجل جديد بأمر APPEND BLANK في نفس الوقت الذي يحاول فيه مستفيد آخر في محطة أخرى استخدام نفس الأمر.

إضافة السجلات

إذا حاولت إضافة سجل جديد في نهاية الملف باستخدام أمر APPEND BLANK وكان الملف مغلقا بواسطة مستفيد آخر في محطة أخرى أو حاول مستفيد آخر في نفس اللحظة استخدام نفس الأمر. فلن ينجح الأمر.

ولأن أمر APPEND BLANK لا يعطي أي مؤشر يفيد نجاحه من عدمه فإننا نلجأ لاستخدام الوظيفة NETERR() للتحقق من نجاح الأمر من عدمه. وتشتمل الوظيفة NETERR() على القيمة .F. إذا نجح الأمر وتم إضافة السجل في نهاية الملف. أما إذا لم ينجح الأمر فإنها تشتمل على القيمة .T.

توجيه مخرجات الطابعة

يمكن التحكم في توجيه المخرجات إلى طابعة معينة في حالة وجود أكثر من طابعة بشبكة الاتصالات وذلك باستخدام أمر

SET PRINTER TO <destination>

ومن هذا الاستعراض للأوامر والوظائف الخاصة بالبرمجة لشبكة الاتصالات يتبين لنا أن «كلبر» تقدم إمكانيات كثيرة لتجنب مشاكل تعديل الملفات أو السجلات من قبل أكثر من مستفيد في نفس الوقت يمكن تلخيصها فيما يلي:

- إمكانية فتح الملف منفردا أو مع آخرين وذلك باستخدام أحد أمرين:

SET EXCLUSIVE

USE <filename> EXCLUSIVE

- إمكانية غلق الملف لتمنع أكثر من مستفيد من تعديل محتويات نفس الملف في نفس اللحظة باستخدام وظيفة FLOCK()
- إمكانية غلق السجل لتمنع أكثر من مستفيد من تعديل محتويات نفس السجل في نفس اللحظة باستخدام وظيفة RLOCK()
- توجيه المخرجات إلى طابعة معينة باستخدام أمر

SET PRINTER TO

- إمكانية التعرف على حالة ملف أو سجل هل هو مغلق أو متاح وذلك من خلال وظيفتين هما:

RLOCK() - FLOCK()

مواجهة مشاكل غلق الملفات والسجلات

عند تصميم نظم لادارة قواعد البيانات لأكثر من مستفيد يجب مراعاة متى يجب استخدام الملف أو السجل بصفة فردية (Exclusive)، ومتى يمكن استخدامه بصفة جماعية (Shared). وذلك لتجنب المشاكل التي قد تنجم من تعديل الملف أو السجل في نفس اللحظة من قبل أكثر من مستفيد.

ونوضح فيما يلي الأمور التي يجب مراعاتها لتجنب هذه المشاكل:

أولا : يجب غلق الملف قبل استخدام الأوامر التي تكتب عليه أو تعدل في محتوياته وهي :

APPEND FROM	RECALL
DELETE	REPLACE
UPDATE	@...GET <fieldname>

لاحظ أن بعض هذه الأوامر تستخدم لتعديل سجل واحد أو لتعديل أكثر من سجل مثل أمر DELETE وأمر RECALL وأمر REPLACE وأمر GET...@. فإذا كان أحد هذه الأوامر يتطلب حذف سجل واحد أو استرجاعه أو تعديله أو استبدال محتوياته فيجب غلق السجل فقط. وإليك بعض الأمثلة للأوامر التي تستخدم سجلا واحدا.

```
DELETE RECORD 5
RECALL RECORD 5
REPLACE FIELD1 WITH FIELD1+5
@5,5 GET FIELD1
```

ثانيا: ينصح بغلق الملف قبل استخدام الأوامر التي تنفذ على كل سجلات الملف والتي تستدعي تعديل محتوياته أو الكتابة عليه. وذلك لتضمن أن البيانات والنتائج لن تتغير أثناء تنفيذ هذه الأوامر بواسطة مستفيد آخر وهذه الأوامر مثل:

SUM - AVERAGE - COUNT - TOTAL ON

ثالثا: بعض الأوامر لا يمكن استخدامها إلا استخداما منفردا مثل

PACK - ZAP - REINDEX

لأنها ذات تأثير على الملف كله ولأن الملف يولد بعدها بشكل جديد. ولذلك فإذا حاولت استخدامها استخداما جماعيا (Shared) فستحصل على رسالة خطأ بهذا الشكل:

System error not exclusive

رابعا: ليس هناك ضرورة لغلق الملف أو السجل قبل استخدام الأوامر التي تقرأ بيانات الملفات فقط. وينصح باستخدامها استخداما جماعيا وهذه الأوامر مثل

LIST - DISPLAY - REPORT FORM

استخدام وظائف خاصة للتغلب على مشاكل البرمجة

ليس المهم فقط أن تعرف متى تغلق الملف أو السجل قبل استخدامه للكتابة عليه أو تعديله. المهم أيضا أن تعرف ماذا يجب أن تفعل إذا لم يغلق الملف أو السجل فقد يكون الملف أو السجل مغلقا بواسطة مستفيد آخر وفي هذه الحالة لن يتم غلقه مباشرة بناء على طلب البرنامج فما هو الحل في هذه الحالة...؟ هل تخرج من النظام وتظهر رسالة للمستفيد تفيد أن خطأ ما قد حدث. أو هل تعيد المحاولة لمرة أو أكثر حتى يتم غلق الملف أو السجل. وذلك إذا كنت تعرف أن الملف أو السجل يستخدم لفترة قصيرة بواسطة مستفيد آخر أو بواسطة برنامج آخر. أو هل تعيد المحاولة لعدد غير محدود من المرات حتى يتم غلق الملف أو السجل.

وتشتمل حزمة «كلبر» على ملف اسمه LOCKS.PRG يشتمل على ٤ وظائف خاصة[1] (UDFS) تستخدم فقط مع تطبيقات شبكة الاتصالات. وهذه الوظائف هي:

١) الوظيفة NET_USE()
٢) الوظيفة FIL_LOCK
٣) الوظيفة REC_LOCK
٤) الوظيفة ADD_REC()

وكل من هذه الوظائف الخاصة تعيد محاولة غلق الملف أو السجل في حالة فشلها لمدة معينة وتفترض هذه الوظائف جميعا أن أمر SET EXCLUSIVE في الوضع OFF وفيما يلي سنستعرض هذه الوظائف الخاصة.

١ فتح الملف باستخدام الوظيفة *NET_USE()*

تستخدم هذه الوظيفة الخاصة إذا أردت إعادة محاولة فتح الملف فرديا أو جماعيا. وقبل استعراض الوظيفة نورد مثالا يوضح كيفية فتح الملف للاستخدام الجماعي (Shared) بدون هذه الوظيفة (انظر شكل ١-١١)

١) راجع الفصل الخامس إذا لم تكن تعرف ما هي الوظائف الخاصة.

```
SET EXCLUSIVE OFF          && اجعل الملفات التالية مشاعة
USE EMPLOYEE
IF NETERR()                && اذا حدث خطأ اثناء فتح الملف
   ? "Network error... File not available"
   CLEAR
   CLEAR ALL
   RETURN
ENDIF
SET INDEX TO EMP_ID
```

شكل ١ـ١١

وتلاحظ في هذا المثال:

* أننـا استخدمنا أمر SET EXCLUSIVE OFF لتفهـم «كلبر» أن الملف/الملفات التالية ستستخدم جماعيا مع شبكة الاتصالات.
* استخدمنا الوظيفة ()NETERROR عقب أمر USE مباشرة لنتأكد هل الملف فتح أم لا.
* لم نستخدم أمر USE ... INDEX في هذه الحالة لأننا لا نعرف هل سيفتح الملف بنجـاح أم لا. ولذلك فإن أمر SET INDEX TO سينفذ فقط إذا فتح الملف بنجاح وننصحك باستخدام هذه الطريقة عند فتح ملفاتك في برامج شبكة الاتصالات.

استخدام الوظيفة الخاصة ()NET_USE

المثال الموجود في شكل ٢ـ١١ يؤدي نفس الوظيفة التي يقوم بها المثال السابق (شكل ١ـ١١). والفرق بينهما أن هذا المثال يستخدم الوظيفة الخاصة ()NET_USE المـوجـودة ضمن ملف LOCKS.PRG (سنشرح هذه الوظيفة بعد قليل). وننصح باستخـدام هذه الوظيفة عند فتح الملفات في برامج شبكة الاتصالات أو تعديلها حسب طلبك.

```
SET EXCLUSIVE OFF          && اجعل الملفات التالية مشاعة
IF NET_USE ("EMPLOYEE",.T.,5)
   SET INDEX TO EMP_ID
ELSE
   ? "Network error... File not available"
   CLEAR
   CLEAR ALL
   RETURN
ENDIF
```

شكل ٢ـ١١

وفي هذا المثال استخدمنا أمر SET INDEX بعد التأكد من نجاح أمر USE المتضمَّن في الوظيفة ()NET_USE

ويشتمـل شكـل ٣ـ١١ على النص الكـامل للوظيفة ()NET_USE كما هي موجودة بملف LOCKS.PRG.

ومن هذا الشكل يتضح الآتي:

* تشتمل الوظيفة دائما على القيمة المنطقية .T. إذا فتح الملف بنجاح فقط أما إذا تعذر فتح الملف فإنها ستشتمل على القيمة المنطقية .F.
* تحاول الوظيفة فتح الملف للاستخدام الفردي أو الجماعي باستخدام ٣ معطيات (parameters) هي :

١ ـ file: اسم ملف قاعدة البيانات المطلوب فتحه.

٢ ـ exuse: طريقة فتح الملف ـ منفردا (.T.) أو بالمشاركة (.F.) ـ

٣ ـ Wait: رقم يوضح عدد الثواني التي سيتم انتظارها حتى يفتح الملف.

وهذه المعطيات تم العويض عنها في البرنامج الذي يستخدم الوظيفة كما يلي:

١ ـ اسم الملف: EMPLOYEE

٢ ـ طريقة فتح المف: .F. وتعني فتح الملف بالمشاركة.

٣ ـ عدد الثواني:. 5

```
****
*  NET_USE function
*
*  Trys to open a file for exclusive or shared use.
*  SET INDEXes in calling procedure if successful.
*  Pass the following parameters
*    1. Character - name of the .DBF file to open
*    2. Logical - mode of open (exclusive/.NOT. exclusive)
*    3. Numeric - seconds to wait (0 = wait forever)
*
*  Example:
*    IF NET_USE("Accounts", .T., 5)
*       SET INDEX TO Name
*    ELSE
*       ? "Account file not available"
*    ENDIF

FUNCTION NET_USE
PARAMETERS file, ex_use, wait
PRIVATE forever

forever = (wait = 0)
DO WHILE (forever .OR. wait > 0)

   IF ex_use                    && exclusive
      USE &file EXCLUSIVE
   ELSE
      USE &file                 && shared
   ENDIF

   IF .NOT. NETERR()            && USE succeeds
      RETURN (.T.)
   ENDIF

   INKEY(1)                     && wait 1 second
   wait = wait - 1
ENDDO
RETURN (.F.)                    && USE fails
* End - NET_USE
```

شكل ٣-١١

٢ إضافة سجل في نهاية الملف باستخدام الوظيفة ()ADD_REC

عندما ترغب في إضافة سجل في نهاية الملف المفتوح باستخدام أمر APPEND BLANK يجب أن تتأكد أن الملف غير مغلق من قبل مستفيد آخر داخل الشبكة، وأن أحداً غيرك لا يحاول في نفس اللحظة إضافة سجل خال في نهاية الملف. وعادة يتسبب أمر APPEND BLANK في إضافة سجل خال في نهاية الملف وفي غلق السجل الجديد مثلما تفعل الوظيفة ()RLOCK.

ويشتمل شكل ٤-١١ على النص الكامل للوظيفة ()ADD_REC كما هي موجودة بملف LOCKS.PRG. وتستخدم هذه الوظيفة لاضافة سجل في نهاية الملف المفتوح. وتشتمل دائما على القيمة المنطقية .T. في حالة إضافة السجل في نهاية الملف بنجاح. أما إذا لم تتمكن «كلبر» من غلق السجل وإضافة سجل في نهاية الملف خلال المدة الزمنية المحددة لاعادة المحاولة فإنها ستشتمل على القيمة المنطقية .F.

وتقبل الوظيفة رقما من خارجها يمثل عدد الثواني التي ستستمر «كلبر» خلالها في محاولة إضافة السجل قبل أن تتلقى القيمة المنطقية .F. للدلالة على أن عملية الاضافة فشلت. وعادة تفشل «كلبر» في إضافة السجل إذا كان الملف أو السجل مغلقا من قبل مستفيد آخر داخل الشبكة أو كان مستفيد آخر يحاول نفس المحاولة في نفس اللحظة.

فإذا حاول أكثر من مستفيد إضافة سجل في نفس اللحظة فإن أحدهم سينفذ فقط لأن هذا الأمر ينفذ بسرعة شديدة وسيتم تعليق الآخر حتى تحاول «كلبر» في المرة أو المرات القادمة فيتم إضافته.

٣ غلق السجل باستخدام الوظيفة ()REC_LOCK

تستخدم هذه الوظيفة لمحاولة غلق السجل الحالي الذي يقف عنده المؤشر داخل الملف المفتوح في المنطقة المختارة. فإذا تم إغلاق السجل بنجاح فلن يسمح لمستفيد آخر داخل الشبكة بتعديل محتويات هذا السجل حتى يُطلق سراحه مرة ثانية.

```
****
*   ADD_REC function
*
*  Returns true if record appended.  The new record is current
*  and locked.
*  Pass the following parameter
*    1. Numeric - seconds to wait (0 = wait forever)
*

FUNCTION ADD_REC
PARAMETERS wait
PRIVATE forever

APPEND BLANK
IF .NOT. NETERR()
   RETURN (.T.)
ENDIF

forever = (wait = 0)
DO WHILE (forever .OR. wait > 0)

   APPEND BLANK
   IF .NOT. NETERR()
      RETURN .T.
   ENDIF

   INKEY(.5)                    && wait 1/2 second
   wait = wait - .5

ENDDO
RETURN (.F.)                    && not locked
* End ADD_REC

* EOF - Locks.prg
```

شكل ٤-١١

ويجب غلق السجل باستخدام هذه الوظيفة قبل تعديل الحقول التي يشتمل عليها. فإذا أراد مستفيد آخر تعديل محتويات نفس السجل بعد ذلك فإن السجل

سيظهر بعد التعديلات التي أجريت عليه. وبالتالي فإن المعلومات التي ستظهر على الشاشة ستعكس البيانات الحقيقية للسجل دائما.

ويشتمل شكل ٥ـ١١ على النص الكامل للوظيفة REC_LOCK() كما هي موجودة بملف LOCKS.PRG. ومن هذا الشكل تلاحظ أن:

```
* REC_LOCK function
*
* Trys to lock the current record
* Pass the following parameter
*    1. Numeric - seconds to wait (0 = wait forever)
*
* Example:
*    IF REC_LOCK(5)
*       REPCLACE Price WITH newprice
*    ELSE
*       ? "Record not available"
*    ENDIF

FUNCTION REC_LOCK
PARAMETERS wait
PRIVATE forever

IF RLOCK()
   RETURN (.T.)          && locked
ENDIF

forever = (wait = 0)
DO WHILE (forever .OR. wait > 0)

   IF RLOCK()
      RETURN (.T.)               && locked
   ENDIF

   INKEY(.5)                     && wait 1/2 second
   wait = wait - .5

ENDDO
RETURN (.F.)                     && not locked
* End - REC_LOCK
```

شكل ٥ـ١١

* هذه الوظيفة تقبل رقما يمثل عدد الثواني التي ستستمر «كلبر» خلالها في محاولة غلق السجل قبل أن تشتمل على القيمة المنطقية .F.
* تشتمل الوظيفة على القيمة المنطقية .T. إذا نجحت محاولة غلق السجل. أما إذا لم تنجح محاولة غلق السجل خلال المدة الزمنية المحددة لاعادة المحاولة فإنها ستشتمل على القيمة المنطقية .F.

والمثال التالي (انظر شكل ٦ـ١١) يوضح كيفية استخدام هذه الوظيفة بفرض أننا نريد أن تستمر «كلبر» في محاولة غلق السجل لمدة ٣ ثواني. ومعنى جملة IF في هذا المثال: إذا كان السجل مغلقا بعد محاولات تستمر لمدة ٣ ثواني يسمح بتعديل الراتب. وإلا فتظهر رسالة تفيد أن السجل مغلق.

```
IF REC_LOCK(3)
  @ 5,5 GET SALARY PICT "99,999.99"
  READ
ELSE
   ? "Network error... Record not available"
ENDIF
```

شكل ٦ـ١١

٤ غلق الملف باستخدام الوظيفة *FIL_LOCK()*

تقوم هذه الوظيفة بنفس عمل الوظيفة REC_LOCK() التي شرحناها قبل قليل. والفرق الوحيد بينهما أن هذه الوظيفة تغلق الملف كله بدلا من سجل واحد داخل الملف.

ومن شكل ٧ـ١١ تلاحظ أن الأوامر التي تشتمل عليها هذه الوظيفة هي نفسها الموجودة في شكل ٥ـ١١ السابق مع فرق واحد وهو استخدام الوظيفة FLOCK() بدلا من الوظيفة RLOCK().

```
****
* FIL_LOCK function
*
* Trys to lock the current shared file
* Pass the following parameter
*    1. Numeric - seconds to wait (0 = wait forever)
*
* Example:
*    IF FIL_LOCK(5)
*       REPLACE ALL Price WITH Price * 1.1
*    ELSE
*       ? "File not available"
*    ENDIF

FUNCTION FIL_LOCK
PARAMETERS wait
PRIVATE forever

IF FLOCK()
   RETURN (.T.)          && locked
ENDIF

forever = (wait = 0)
DO WHILE (forever .OR. wait > 0)

   INKEY(.5)                    && wait 1/2 second
   wait = wait - .5

   IF FLOCK()
      RETURN (.T.)              && locked
   ENDIF

ENDDO
RETURN (.F.)                    && not locked
* End - FIL_LOCK
```

شكل ٧ـ١١

والمثال التالي (انظر شكل ٨ـ١١) يستخدم هذه الوظيفة بفرض أننا نرغب أن تستمر محاولة غلق الملف لمدة ٥ ثواني.

```
IF FIL_LOCK(5)
  TOTAL ON DEPT TO TOTDEPT
ELSE
  ? "Network error... File not available"
  CLEAR ALL
  RETURN
ENDIF
```

شكل ٨-١١

ملاحظة: راجع النظام الموجود في الباب الثالث من هذا الكتاب لتتعرف على وظائف أخرى تتحكم في فتح وغلق الملفات وفي غلق السجلات قبل تعديلها. وهي ذات فائدة كبرى في إعداد نظم إدارة قواعد بيانات تستخدم مع شبكة اتصالات محلية. لأن النظام معد بطريقتين. الاستخدام المنفرد والاستخدام مع شبكة اتصالات محلية.

تـذكـر...!

عند تطوير نظم إدارة قواعد بيانات لخدمة أكثر من مستفيد يجب مراعاة الآتي:

- غلق السجل قبل استخدام الأوامر التي ستعدل في محتويات السجل الواحد.
- غلق الملف قبل استخدام الأوامر التي ستعدل محتويات الملف كله.
- استخـدام الملف استخداما منفردا (exclusive) في حالة استخدام أكثر من أمر لتغيير محتويات الملف.

الباب الثالث

تطبيقات شاملة

الفصل الثاني عشر

تطبيقات شاملة

يشتمل هذا الفصل على نظام للمبيعات يعتبر عينة يمكن الاسترشاد بها في إعداد نظم إدارة قواعد بيانات مماثلة. والنظام يشتمل على الوظائف الأساسية التي لا يخلو منها أي نظام لادارة قواعد البيانات وهي:

- صيانة ملفات العملاء بالاضافة أو التعديل أو الحذف.
- إصدار فواتير البيع أو تعديل بيانات فاتورة أصدرت أو حذفها بالكلية.
- استخراج التقارير اللازمة عن العملاء.
- صيانة ملفات النظام مثل: عمل نسخ احتياطية للملفات واسترجاع النسخ المحفوظة وتنظيف الملفات وترتيبها.

تتميز قاعدة البيانات «كلبر» على غيرها من الحزم البراجية التي تستخدم لاعداد نظم إدارة قواعد البيانات بقوتها وإمكانياتها العديدة التي تسهل إعداد النظم وتزيد من قوتها. والنظام الذي سنورده بعد قليل يستخدم معظم الامكانيات والتسهيلات التي تشتمل عليها «كلبر» والنظام يعتمد على إظهار قائمة اختيارات يتبعها قوائم أخرى لتنفيذ الوظائف المطلوبة. مما يسهل استخدام النظام من قبل المستفيد النهائي. ويتسبب كل اختيار من اختيارات القائمة الرئيسية للنظام أو القوائم التابعة لها في استدعاء برنامج أو إجراء مستقل وتنفيذه والهدف من تقسيم النظام إلى أكثر من برنامج والبرنامج إلى إجراءات (في بعض الأحيان) أن تكون هذه البرامج صغيرة ليسهل عليك فهمها وتعديلها لتناسب حاجتك الخاصة في نظم إدارة قواعد البيانات الأخرى.

وننصح باستخدام هذه الطريقة حتى عند إعداد نظم لادارة قواعد البيانات أكبر أو أشد تعقيدا. وتسمى هذه الطريقة البرمجة التركيبية (modular programming) وقد شرحناها بالتفصيل في كتابها المرجع **الأساسي لقاعدة البيانات dBASE III PLUS**. وتتلخص في تطوير برامج كثيرة صغيرة لتكوّن في النهاية نظاما متكاملا. وبعد تطوير كل برنامج منها واختباره منفردا يتم تطوير البرنامج الرئيسي (القائمة الرئيسية) ليربط هذه البرامج مع بعضها في نظام واحد.

وفي النهاية ستكون قادرا بإذن الله على تصميم وتطوير نظم أخرى لتؤدي نفس الوظائف أو أكثر منها في نظم إدارة قواعد البيانات الأكثر تعقيدا.

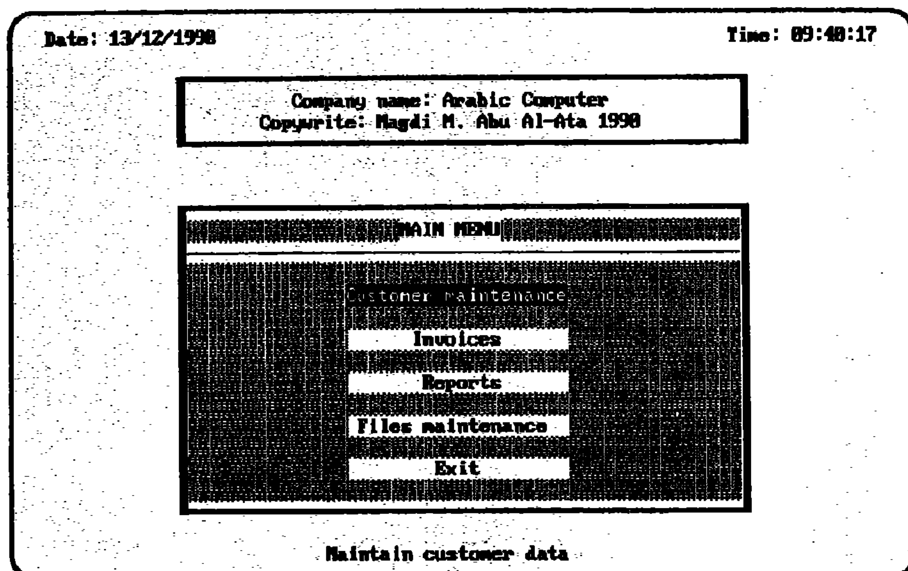
وسنورد فيما يلي البرامج والاجراءات التي يتكون منها النظام وسنوضح في التعليق على هذه البرامج المفاهيم الهامة أو الغامضة فقط. فإذا وجدت صعوبة في فهم أحد الأوامر أو الوظائف يمكنك الرجوع إلى الباب الرابع. مرجع الأوامر والوظائف. وكما تلاحظ فقد كتبنا أمام كل أمر رقم السطر لسهولة الاشارة إليه أثناء الشرح. وطبعا هذا الرقم ليس جزءا من الأمر ويجب استبعاده عند الرغبة في تنفيذ البرنامج.

القائمة الرئيسية للنظام

تشتمل القائمة الرئيسية للنظام على أربعة اختيارات رئيسية تسمح بأداء الوظائف الرئيسية التالية:

١ ـ صيانة ملفات العملاء بالاضافة أو الحذف أو التعديل.
٢ ـ إصدار فواتير البيع أو تعديلها أو حذفها.
٣ ـ إعداد التقارير اللازمة.
٤ ـ صيانة ملفات النظام مثل عمل نسخ احتياطية للملفات واسترجاع النسخ المحفوظة وتنظيف الملفات وترتيبها.

ويشتمل شكل ١ـ١٢ على القائمة المقترحة. ويتم الانتقال من اختيار إلى آخر داخل هذه القـائمـة باستخـدام مفـاتيح الأسهم ↓ أو ↑ كما يتم تنفيذ أحد هذه الاختيارات بتحريك المؤشر إلى الاختيار المطلوب ثم ضغط مفتاح الادخال.



شكل ١ ـ ١٢ القائمة الرئيسية للنظام

ويشتمل شكل ٢ـ١٢ على البرنامج اللازم لاظهار القائمة الرئيسية للنظام وتنفيذ اختياراتها. وعن هذا البرنامج نوضح ما يلي:

ـ جملة IF...ENDIF (سطر ١٣ـ١٧) تسأل هل تم إدخال معطيات (paramet- ers) من محث نظام التشغيل DOS عنـد استدعاء النظام أم لا؟ فإذا أدخل

```
 1 * ------------------------------------------------------------------*
 2 * Program    : SAMAIN.PRG                                          *
 3 * Author     : Magdi M. Abu Al-Ata                                 *
 4 * Date       : October 24, 1990                                    *
 5 * Purpose    : Main Module for main menu                           *
 6 * Copyright  : Magdi M. Abu Al-Ata                                 *
 7 * Parameters : From DOS, you can optionally pass any parameter     *
 8 *              to run in color mode.                               *
 9 *              missing a parameter will run in  monochrome mode    *
10 * ------------------------------------------------------------------*
11 PUBLIC hd1,hd2
12 * Determine whether the user wants to use color or monochrome monitor
13 IF PCOUNT() > 0
14 |    PUBLIC syscolor
15 |    syscolor = "W+/B+,GR+/R+"
16 |    SETCOLOR(syscolor)
17 ENDIF
18 *  Set clipper environment
19 SET SCOREBOARD OFF
20 SET WRAP ON
21 SET ESCAPE OFF
22 SET MESSAGE TO 24 CENTER
23 SET DELETED ON
24 SET CONFIRM ON
25 SET DATE BRITISH
26 SET CENTURY ON
27 * Save the DOS screen
28 SAVE SCREEN TO dos_scr
29 CLEAR
30 dubl = CHR(201)+CHR(205)+CHR(187)+CHR(186) ;
30  +CHR(188)+CHR(205)+CHR(200)+ CHR(186)+CHR(176)
31 @  2, 14 TO 5, 66 DOUBLE
32 hd1 =  "Company name: Arabic Computer"
33 hd2 =  "Copywrite: Magdi M. Abu Al-Ata 1990"
34 @ 3,t_cent(hd1,80) SAY hd1
35 @ 4,t_cent(hd2,80) SAY hd2
36 @ 0,0
37 @ 0,2  SAY "Date: "+ DTOC(DATE())
38 @ 0,65 SAY "Time: "+ TIME()
39
40 @  8, 14, 22, 66 BOX dubl
41 @  9,t_cent("MAIN MENU",80) SAY "MAIN MENU"
42 @ 10, 15 TO 10, 65
43 DO WHILE .T.
44 |     @ 12,30 PROMPT "Customer maintenance" ;
44 |       MESSAGFE "Maintain customer data "
45 |     @ 14,30 PROMPT "      Invoices       " ;
45 |       MESSAGE "Add, Edit, and Delete invoices"
46 |     @ 16,30 PROMPT "      Reports        " ;
46 |       MESSAGE "Send reports to the printer"
47 |     @ 18,30 PROMPT " Files maintenance  " ;
47 |       MESSAGE "Index, Back up, Restore and Pack files"
```

شكل ٢ - ١٢ برنامج SAMAIN. PRG.

```
48 |     @ 20,30 PROMPT "        Exit
48 |       MESSAGE "Quit to DOS"
49 |     MENU TO choice1
50 |     DO CASE
51 |     CASE choice1 = 1
52 |     |    SAVE SCREEN TO menu_scr
53 |     |    menu_clr = SETCOLOR ()
54 |     |    @24,0 CLEAR
55 |     |    DO sacust
56 |     |    SETCOLOR (menu_clr)
57 |     |    RESTORE SCREEN FROM menu_scr
58 |     CASE choice1 = 2
59 |     |    SAVE SCREEN TO menu_scr
60 |     |    menu_clr = SETCOLOR()
61 |     |    @24,0 CLEAR
62 |     |    DO sainv
63 |     |    SETCOLOR (menu_clr)
64 |     |    RESTORE SCREEN FROM menu_scr
65 |     CASE choice1 = 3
66 |     |    SAVE SCREEN TO menu_scr
67 |     |    menu_clr = SETCOLOR ()
68 |     |    @24,0 CLEAR
69 |     |    DO sarep
70 |     |    SETCOLOR (menu_clr)
71 |     |    RESTORE SCREEN FROM menu_scr
72 |     CASE choice1 = 4
73 |     |    SAVE SCREEN TO menu_scr
74 |     |    menu_clr = SETCOLOR()
75 |     |    @24,0 CLEAR
76 |     |    DO sautil
77.|     |    SETCOLOR (menu_clr)
78 |     |    RESTORE SCREEN FROM menu_scr
79 |     CASE choice1 = 5
80 |<-------EXIT
81 |     ENDCASE
82 ENDDO
83 RESTORE SCREEN FROM dos_scr
84 SET COLOR TO
85 QUIT
86
87 ************************
88 *  End of main program  *
89 ************************
90
91 *-------------------------------*
92 *   Procedures and Functions     *
93 *   Called only by SAMAIN.PRG   *
94 *-------------------------------*
95
96 ****************
97 *  Function      : t_cent()
98 *  Purpose       : Displays text centered on the screen
99 *  Syntax        : t_cent (tstring, tlen)
```

تابع شكل ٢ ـ ١٢

```
100 *  Parameters     :
101 *      tstring    : Text to be displayed
102 *      tlen       : the width you need text to be centered on
103 *  Example        : t_cent( "Clipper world!", 40 )
104 ****************
105 FUNCTION t_cent
106 PRIVATE tstring, tlen
107 PARAMETERS tstring, tlen
108 STORE INT(tlen/2 - (LEN(tstring)/2)) TO bigcol
109 RETURN bigcol
110
111 ****************************
112 *  End of file SAMAIN.PRG   *
113 ****************************
```

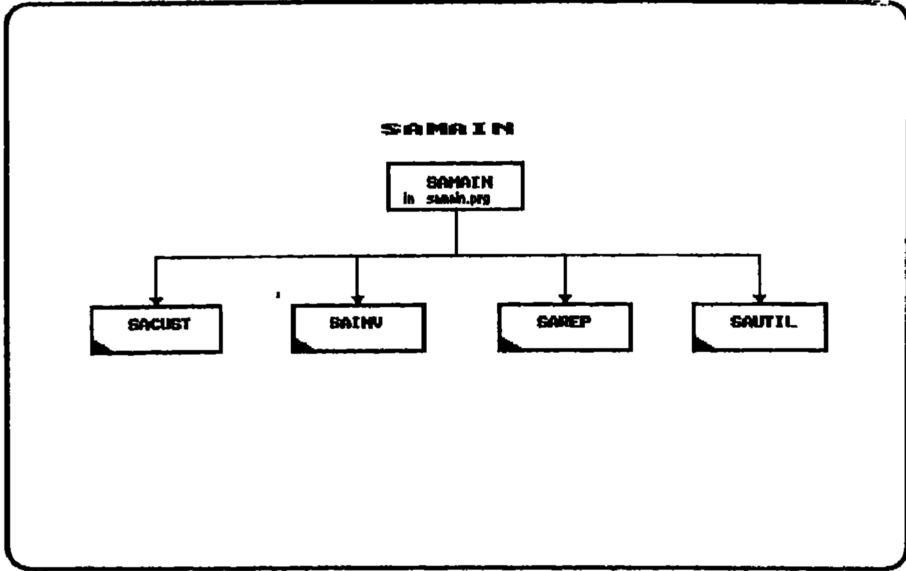
تابع شكل ٢ - ١٢

المستفيد واحدة أو أكثر فإن النظام سيعمل بالألوان المحددة. أما إذا لم يدخل أي معطيات مع اسم النظام فسيعمل النظام بلونين فقط أبيض وأسود. فمثلا: SAMAIN تتسبب في تشغيل النظام بدون ألوان أما SAMIN C تتسبب في تشغيل النظام بالألوان المختارة.

- يتم حفظ الشاشة المعروضة قبل تشغيل النظام (سطر رقم ٢٨) واسترجاعها عند الانتهاء منه (سطر رقم ٨٣).
- يتم حفظ القائمة الرئيسية للنظام قبل استدعاء قائمة فرعية (سطر رقم ٥٢) واسترجاعها بعد الانتهاء من القائمة الفرعية (سطر رقم ٥٧).
- يستخدم البرنامج وظيفة خاصة t_centr(). ومهمتها ضبط عبارة وسط مساحة معينة على الشاشة وقد وضعناها في نهاية البرنامج أما العلاقة بين البرنامج الرئيسي في النظام والبرامج الأربعة التي تستدعي القوائم الفرعية فيوضحها شكل ٣-١٢.

قائمة صيانة ملفات العملاء

يمثل كل اختيار من اختيارات القائمة الرئيسية الموجودة في شكل ١-١٢ وظيفة من وظائف إدارة قاعدة البيانات وتشتمل كل وظيفة على أكثر من عمل أو وظيفة فرعية داخلها. ويشار للأعمال أو الوظائف الفرعية التي تتفرع عن الاختيارات الرئيسية



شكل ٣ - ١٢ خريطة برنامج SAMAIN. PRG

بالمستوى الثاني داخل النظام. فمثلا برنامج CUST.PRG يتم استدعاؤه من القائمة الرئيسية للنظام ويشتمل على الأعمال والوظائف التي تمكننا من صيانة ملفات العملاء وتشمل:

١ - إضافة عميل جديد إلى ملف العملاء (Add)

٢ - تعديل بيانات عميل موجود (Edit)

٣ - حذف بيانات عميل موجود (Delete)

ويوضح شكل ٤-١٢ قائمة صيانة ملفات العملاء وهي تستخدم في حالات الاضافة أو التعديل أو الحذف. أما البرنامج اللازم لاظهار هذه القائمة وتنفيذ اختياراتها فتجده في شكل ٥-١٢ وعن هذا البرنامج نوضح ما يلي:

- هذا البرنامج شبيه بالبرنامج الموجود في شكل ٢-١٢ وهو يستخدم قائمة اختيارات أفقية بدلا من القائمة الرأسية الموجودة في شكل ٢-١٢ .
- يشتمل البرنامج على الاجراء cust_fmt وهذا الاجراء خاص بإظهار أسماء الحقول داخل سجل العميل على الشاشة تمهيدا لاضافتها أو تعديلها أو

```
Date: 13/12/1990                                              Time: 09:40:17
                Add        Edit        Delete      Return

                          Customer main data

              Account No.   :
              Customer name:
              Company Name  :
              Address       :
              City          :
              Phone         :

                    Add a new customer to the file
```

شكل ٤ ـ ١٢ قائمة صيانة ملفات العملاء

```
1 * ---------------------------------------------------------------*
2 * Program    : SACUST.PRG                                         *
3 * Author     : Magdi M. Abu Al-Ata                                *
4 * Date       : October 24, 1990                                   *
5 * Purpose    : To paint customer menu                             *
6 * Copyright  : Magdi M. Abu Al-Ata                                *
7 * Called from: SAMAIN.PRG                                         *
8 * ---------------------------------------------------------------*
9 @ 1,0 CLEAR TO 24,79
10 DO WHILE .T.
11 |    @ 1,14 TO 3,66 DOUBLE
12 |    @ 6,10 TO 22,70 DOUBLE
13 |    @ 7,25 SAY "Customer main data"
14 |    @ 8,11 TO 8,69
15 |    DO cust_fmt
16 |    @ 2,20 PROMPT "Add" MESSAGE "Add a new customer to the file"
17 |    @ 2,29 PROMPT "Edit" MESSAGE "Modify customer record"
18 |    @ 2,41 PROMPT "Delete"  MESSAGE "Delete a customer record"
19 |    @ 2,52 PROMPT "Return"  MESSAGE "Return to the main menu"
20 |    MENU TO choice3
21 |    DO CASE
22 |    CASE choice3 = 0
```

شكل ٥ ـ ١٢ برنامج SACUST. PRG

```
23 <--------RETURN
24 |    CASE choice3 = 1
25 |   |    DO sacustad
26 |    CASE choice3 = 2
27 |   |    DO sacusted
28 |    CASE choice3 = 3
29 |   |    DO sacustde
30 |    CASE choice3 = 4
31 <--------RETURN
32 |    ENDCASE
33 ENDDO
34
35 *************************
36 *  End of program       *
37 *************************
38
39 *--------------------------------*
40 *   Procedures and Functions     *
41 *   Called only by SACUST.PRG    *
42 *--------------------------------*
43
44 ****************
45 *  Procedure   : cust_fmt
46 *  Purpose     : Drows customer screen format
47 ****************
48 PROCEDURE cust_fmt
49 * Make the variables globally available to the subroutines.
50 PUBLIC m_acct, m_company, m_addr, m_fnam, m_mnam, m_lnam, m_city,  m_phone
51 STORE SPACE (4) to m_acct
52 STORE SPACE (20) to m_company, m_addr
53 STORE SPACE (12) to m_fnam, m_mnam, m_lnam
54 STORE SPACE (17) to m_city
55 STORE SPACE (8) to m_phone
56 @ 10, 12 SAY "Account No.  :"
57 @ 10, 27 SAY m_acct
58 @ 12, 12 SAY "Customer name:"
59 @ 12, 27 GET m_fnam
60 @ 12, 41 GET m_mnam
61 @ 12, 54 GET m_lnam
62 @ 14, 12 SAY "Company Name :" GET m_company
63 @ 16, 12 SAY "Address      :" GET m_addr
64 @ 18, 12 SAY "City         :" GET m_city
65 @ 20, 12 SAY "Phone        :" GET m_phone PICTURE "999-9999"
66 CLEAR GETS
67 RETURN
68
69 *****************************
70 *  End of file SACUST.PRG   *
71 *****************************
```

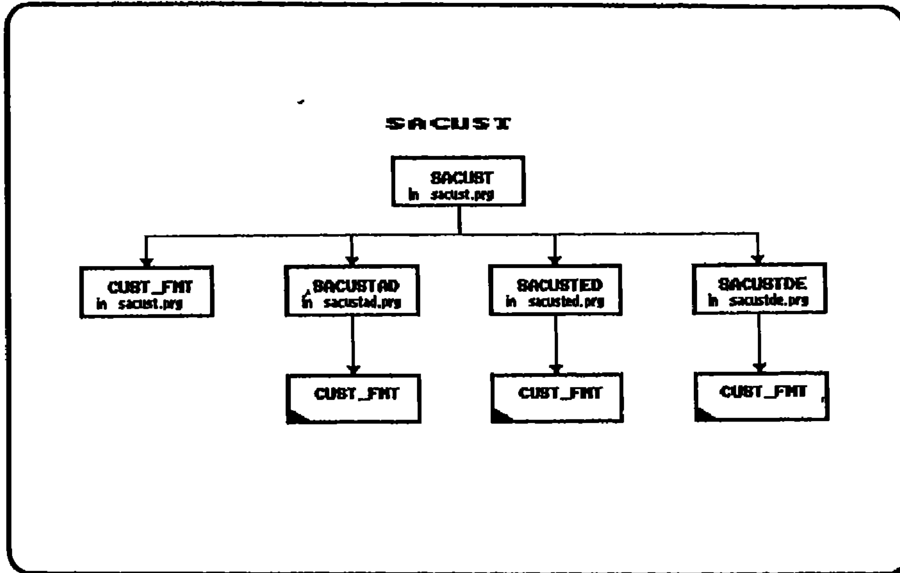
تابع شكل ٥ - ١٢

حذفهـا. وقـد وضعنا في هذا الاجراء في آخر البرنامج ومن المناسب وضع الأوامر التي يشتمل عليها داخل إجراء مستقل لأن هذا الاجراء سيستخدم مع كل البرامج التي تتولى صيانة ملف العملاء (الاضافة والحذف والتعديل).
يوضح شكل ١٢ـ٦ العلاقة بين برنامج SACUST.PRG والبرامج التي يستدعيها.

ونوضح فيما يلي البرامج الثلاثة التي تستخدم لصيانة ملف العملاء.

برنامج الاضافة *SACUSTAD.PRG*

يوضح شكل ١٢ـ٧ برنامج SACUSTAD.PRG الذي يستخدم لاضافة عميل جديد إلى الملف كما يوضح شكل ١٢ـ٨ العلاقة بين هذا البرنامج والبرامج الأخرى المرتبطة به داخل النظام. وعن هذا البرنامج نوضح ما يلي:
- يبدأ البرنامج بفتح ملف العملاء (CUSTMR.DBF) وملف الفهرس (CUST.NTX). ويشتمل شكل ١٢ـ٩ على مواصفات ملف



شكل ٦ - ١٢ خريطة برنامج SACUST. PRG

```
1 * ----------------------------------------------------------------*
2 * Program    : SACUSTAD.PRG                                        *
3 * Author     : Magdi M. Abu Al-Ata                                 *
4 * Date       : October 24, 1990                                    *
5 * Purpose    : Adds new customer                                   *
6 * Copyright  : Magdi M. Abu Al-Ata                                 *
7 * Called from: SACUST.PRG                                          *
8 * Called from: SAMAIN.PRG                                          *
9 * ----------------------------------------------------------------*
10 @24,0 CLEAR
11 USE custmr
12 SET INDEX TO custmr
13 DO WHILE .T.
14 |     SET ESCAPE ON
15 |     @ 24,24 SAY "Press Esc to return to the menu"
16 |     @ 10,27 GET m_acct PICTURE '9999' VALID new_rec()
17 |     READ
18 |     @ 24,0
19 |     IF LASTKEY() = 27
20 |     |    CLOSE DATABASES
21 |     |    @ 24, 0
22 <---------RETURN
23 |     ENDIF
24 |     SET ESCAPE OFF
25 |     @ 12, 27 GET m_fnam
26 |     @ 12, 41 GET m_mnam
27 |     @ 12, 54 GET m_lnam
28 |     @ 14, 27 GET m_company
29 |     @ 16, 27 GET m_addr
30 |     @ 18, 27 GET m_city
31 |     @ 20, 27 GET m_phone PICTURE "@K 999-9999"
32 |     READ
33 |     REPLACE company with m_company, fnam with m_fnam, ;
33 |      mnam with m_mnam, lnam with m_lnam, addr with m_addr, ;
33 |      city with m_city , phone with m_phone
34 |     STORE " " TO m_ok
35 |     @ 24, 20 SAY "Enter another new customer? (Y/N)---) "GET m_ok
36 |     READ
37 |     IF UPPER (m_ok) = "Y"
38 |     |    @ 24,0 CLEAR
39 |     |    DO cust_fmt
40 |     ELSE
41 |     |    CLEAR
42 |     |    CLOSE DATABASES
43 <---------RETURN
44 |     ENDIF
45 ENDDO
46
47 ************************
48 *  End of program      *
49 ************************
50
51 *------------------------------*
```

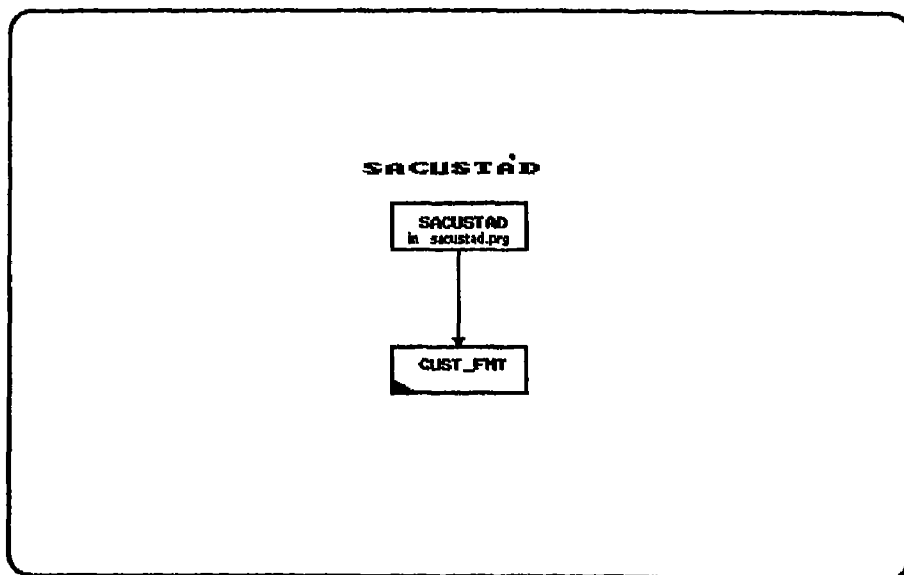
شكل ٧ ـ ١٢ .برنامج SACUSTAD. PRG

```
52 *   Procedures and Functions      *
53 *   Called only by SACUSTAD.PRG   *
54 *--------------------------------*
55
56 ****************
57 *  Function     : new_rec()
58 *  Purpose      : To confirm that the account no. not exist
59 *  Syntax       : new_rec()
60 *  Returns      : .T. or .F.
61 *  Example      : IF new_rec()
62 ****************
63 FUNCTION new_rec
64 IF m_acct = SPACE(4)
65 |     TONE (349,3)
66 |     TONE (300,5)
67 |     @ 24,0
68 |     @ 24,25 SAY "Press Esc to go back to menu."
69 <----RETURN .F.
70 ENDIF
71 SEEK &m_acct
72 IF .NOT. EOF()
73 |     TONE (349,3)
74 |     @ 24,0
75 |     @ 24,26 SAY "Duplicate Number. Try Again."
76 <----RETURN .F.
77 ELSE
78 |     APPEND BLANK
79 |     @ 24,0
80 |     REPLACE acct with m_acct
81 <----RETURN .T.
82 ENDIF
83
84 ****************************
85 *  End of file SACUSTAD.PRG *
86 ****************************
```

تابع شكل ٧ ـ ١٢

CUSTMR.DBF الـذي يستخـدم مع برامج الاضافة والتعديل والحذف. ويمكنك إنشاء هذا الملف إما باستخدام أمر CREATE CUSTMR من محث «دي بيس ثري بلاس» أو باستخدام ملف DBU.EXE الذي يأتي مع حزمة «كلبر» (راجع الفصل الثامن).

وقـد استخـدمنـا الـطريقـة الأخـيرة لانشاء ملف CUSTMR.DBF والملفات الأخرى التي يشتمل عليها النظام.



شكل ٨ ـ ١٢ خريطة برنامج SACUSTAD. PRG

F1 Help | F2 Open | F3 Create | F4 Save | F5 Browse | F6 Utility | F7 Move | F8 Set

Files

Structure of CUSTMR.DBF Field 8

Field Name	Type	Width	Dec
ACCT	Character	4	
COMPANY	Character	20	
FNAM	Character	12	
MNAM	Character	12	
LNAM	Character	12	
ADDR	Character	20	
CITY	Character	17	
PHONE	Character		

شكل ٩ ـ١٢ـ مواصفات ملف CUSTMR. DBF

- قبل إضافة عميل جديد إلى الملف يتأكد البرنامج أن هذا العميل غير موجود من قبل والمفتاح المستخدم لهذا الغرض هو رقم حساب العميل (ACCT). ويستخدم البرنامج الوظيفة الخاصة mew_rec() لهذا الغرض. وقد وضعنا الوظيفة new_rec() في آخر البرنامج (سطر ٦٣) وهي تقوم بالآتي:

أ ـ إذا أدخل المستخدم فراغات محل رقم الحساب فإنه سيسمع صوت الجرس وتظهر له رسالة (سطر ٦٥ وما بعده).

ب ـ إذا كان رقم الحساب موجودا (سطر رقم ٧٢) سيسمع صوت الجرس وتظهر له رسالة لاعادة المحاولة. وستشتمل الوظيفة على القيمة .F.

جـ ـ إذا كان رقم الحساب غير موجود (سطر رقم ٧٧) بالملف فإن الوظيفة ستشتمل على .T. أي أن الشرط الموجود في السطر رقم ١٦ سيتحقق سجلا خاليا في نهاية الملف وسيحل الرقم الذي أدخل محل الفراغات الموجودة برقم الحساب في السجل المضاف وسينتهي أمر READ. وبالتالي سينفذ باقي البرنامج وستقبل بيانات السجل أي سيضاف سجل جديد للملف.

- بعد الانتهاء من إضافة السجل تظهر رسالة للمستخدم لتتأكد من هل يريد إضافة بيانات عميل آخر أم لا. وبناء على إجابته تتكرر المحاولة أو ينتهي البرنامج.

برنامج الحذف *SACUSTDE.PRG*

يوضح شكل ١٠-١٢ برنامج SACUSTDE.PRG الذي يستخدم لحذف بيانات عميل من الملف. كما يوضح شكل ١١-١٢ العلامة بين هذا البرنامج والبرامج الأخرى المرتبطة به داخل النظام. وعن هذا البرنامج نوضح ما يلي:

- يبدأ البرنامج بفتح ملف العملاء وملف الفهرس (انظر شكل ٩-١٢).
- جملة IF في سطر ١٩ تسمح بالعودة للقائمة في حالة ضغط مفتاح Esc بعد غلق الملفات.
- السطر رقم ١٦ يظهر ٤ فراغات لادخال رقم الحساب ـ الذي يجب أن يكون ٤ أعداد. وقبل حذف بيانات العميل يتأكد البرنامج من أن هذا العميل موجود بالملف والمفتاح المستخدم لهذا الغرض هو رقم حساب العميل

```
 1 * -----------------------------------------------------------------*
 2 * Program     : SACUSTDE.PRG                                       *
 3 * Author      : Magdi M. Abu Al-Ata                                *
 4 * Date        : October 24, 1990                                   *
 5 * Purpose     : To delete an existing customer                     *
 6 * Copyright   : Magdi M. Abu Al-Ata                                *
 7 * Called from: SACUST.PRG                                          *
 8 * Called from: SAMAIN.PRG                                          *
 9 * -----------------------------------------------------------------*
10 @24,0
11 USE CUSTMR
12 SET INDEX TO custmr
13 DO WHILE .T.
14 |    SET ESCAPE ON
15 |    @ 24,24 SAY "Press Esc to return to the menu"
16 |    @ 10,27 GET m_acct PICTURE "@K 9999" VALID rec_fnd()
17 |    READ
18 |    @ 24,0
19 |    IF LASTKEY() = 27
20 |    |    CLOSE DATABASES
21 |    |    @ 24,0
22 <--------RETURN
23 |    ENDIF
24 |    SET ESCAPE OFF
25 |    @ 12,27 GET fnam
26 |    @ 12,41 GET mnam
27 |    @ 12,54 GET lnam
28 |    @ 14,27 GET company
29 |    @ 16,27 GET addr
30 |    @ 18,27 GET city
31 |    @ 20,27 GET phone PICTURE "@K 999-9999"
32 |    CLEAR GETS
33 |    IF .NOT. EOF()
34 |    |    STORE " " TO m_ok
35 |    |    @24, 20 SAY "Delete this customer record? (Y/N)---->" GET m_ok
36 |    |    READ
37 |    |    IF UPPER (m_ok) = "Y"
38 |    |    |    DELETE
39 |    |    |    @24,0
40 |    |    |    DO cust_fmt
41 |    |    |    STORE " " TO m_ok
42 |    |    |    @ 24, 20 SAY "Delete another customer record? (Y/N)----> "GET m_ok
43 |    |    |    READ
44 |    |    |    IF UPPER (m_ok) = "Y"
45 |    |    |    |    @24,0
46 |    |    |    |    DO cust_fmt
47 ^---------------------LOOP
48 |    |    |    ENDIF
49 |    |    ENDIF
50 |    ENDIF
51 |    CLEAR
52 |    CLOSE DATABASES
```

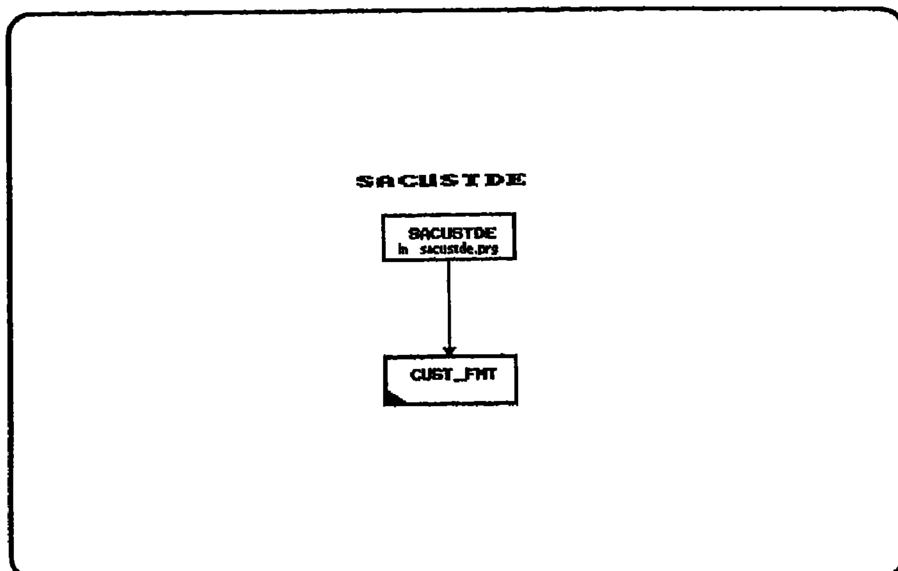
شكل ١٠ ـ ١٢ برنامج SACUSTDE. PRG

```
53 <----RETURN
54 ENDDO
55
56 ************************
57 *  End of program      *
58 ************************
59
60 *--------------------------------*
61 *   Procedures and Functions     *
62 *   Called only by SAMAIN.PRG    *
63 *--------------------------------*
64
65 ****************
66 *  Function        : rec_fnd()
67 *  Purpose         : Validation for edit and delete process
68 *  Returns         : .T. or .F.
69 *  Syntax          : rec_fnd()
70 *  Example         : IF rec_fnd()
71 ****************
72 FUNCTION rec_fnd
73 IF m_acct = SPACE(4)      && Choosing SPACE(4) is okay.
74 |     GO BOTTOM
75 |     SKIP
76 <----RETURN .T.
77 ENDIF
78 SEEK &m_acct
79 IF EOF()
80 |     TONE(349,3)
81 |     @ 24,22 SAY "Invalid account number. Try again."
82 <----RETURN .F.
83 ELSE
84 <----RETURN .T.
85 ENDIF
86
87 ****************************
88 *  End of file SACUSTDE.PRG *
89 ****************************
```

تابع شكل ١٠ - ١٢

ويستخدم البرنامج الوظيفة الخاصة rec_fnd() لهذا الغرض وقد وضعناها في آخر البرنامج (سطر رقم ٧٢) وهي تتأكد من وجود رقم الحساب بالملف قبل الانتقال للأمر التالي في البرنامج أما إذا لم يكن رقم الحساب موجودا بالملف فإنها تظهر رسالة مع الجرس وتسمح بإعادة المحاولة. وهذه الوظيفة تستخدم أيضا مع برنامج SACUSTED.PRG الذي يستخدم لتعديل بيانات العميل بنفس الطريقة.

- السطر رقم ٣٢ يمنع صلاحيات التعديل للحقول التي ظهرت بأمر GET لأن الحالة هنا هي الحذف.



شكل ١١ - ١٢ خريطة برنامج SACUSTDE. PRG

```
 1 * ---------------------------------------------------------------*
 2 * Program     : SACUSTED.PRG                                     *
 3 * Author      : Magdi M. Abu Al-Ata                              *
 4 * Date        : October 24, 1990                                 *
 5 * Purpose     : To edit an existing customer                     *
 6 * Copyright   : Magdi M. Abu Al-Ata                              *
 7 * Called from: SACUST.PRG                                        *
 8 * Called from: SAMAIN.PRG                                        *
 9 * Notes       : It uses rec_fnd() FUNCTION. It is in SACUSTDE.PRG *
10 * ---------------------------------------------------------------*
11 @24,0
12 USE custmr
13 SET INDEX TO custmr
14 DO WHILE .T.
15 |     SET ESCAPE ON
16 |     @ 24,24 SAY "Press Esc to return to the menu"
17 |     @ 10,27 GET m_acct PICTURE "@K 9999" VALID rec_fnd()
18 |     READ
19 |     @ 24,0              && To erase the message
20 |     IF LASTKEY() = 27
21 |     |     CLOSE DATABASES
22 |     |     @ 24, 0
23 <--------RETURN
```

شكل ١٢ - ١٢ برنامج SACUSTED. PRG

```
24 |     ENDIF
25 |     SET ESCAPE OFF
26 |     IF m_acct <> SPACE(4)
27 |     |    @ 12,27 GET fnam
28 |     |    @ 12,41 GET mnam
29 |     |    @ 12,54 GET lnam
30 |     |    @ 14,27 GET company
31 |     |    @ 16,27 GET addr
32 |     |    @ 18,27 GET city
33 |     |    @ 20,27 GET phone PICTURE "@K 999-9999"
34 |     |    READ
35 |     ELSE
36 |<-------EXIT
37 |     ENDIF
38 |     * Determine if the operator wants to edit another.
39 |     STORE " " TO m_ok
40 |     @ 24, 0
41 |     @ 24, 20 SAY "Edit another customer record? (Y/N)----> " GET m_ok
42 |     READ
43 |     IF UPPER (m_ok) = "Y"
44 |     |    @24,0
45 |     |    DO cust_fmt
46 |     ELSE
47 |<-------EXIT
48 |     ENDIF
49 ENDDO
50 CLEAR
51 CLOSE DATABASES
52 RETURN
53
54 ****************************
55 *  End of file SACUSTED.PRG *
56 ****************************
```

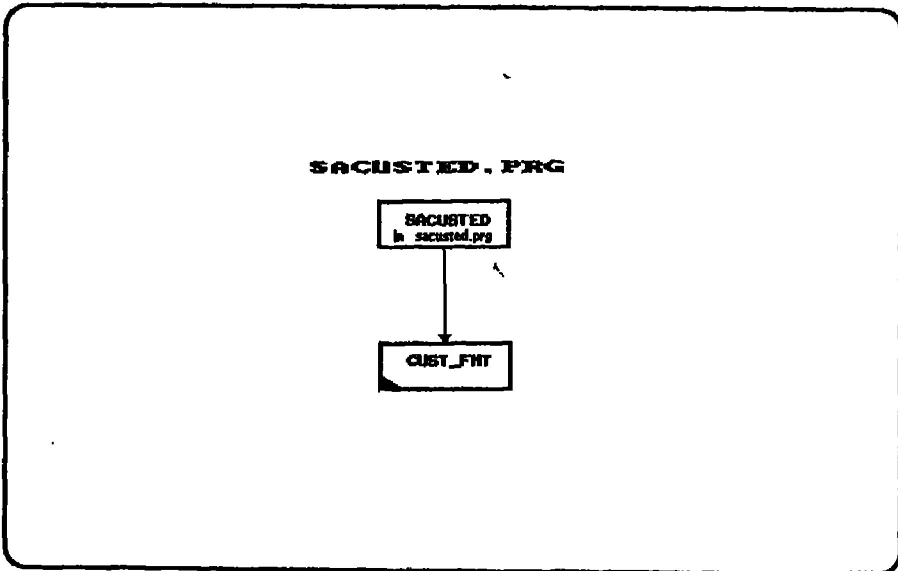
تابع شكل ١٢ - ١٢

- بعد إظهار بيانات العميل تظهر رسالة للمستخدم ليتأكد أن هذا هو السجل المطلوب حذفه. فإذا كانت الاجابة بنعم يتم وضع علامة أمام السجل تدل على أنه مطلوب حذفه وتظهر رسالة أخرى لتعطيه الفرصة لاعادة المحاولة.
- اكتفينا باستخدام أمر DELETE فقط ولم نستخدم أمر PACK لأن الحذف النهائي يستغرق بعض الوقت. ولذلك فإن هذا الاجراء يتم بأحد برامج صيانة الملفات الذي سنشرحه بعد قليل وهو يحذف كل السجلات المعلمة للحذف. وهذه الطريقة تؤدي إلى سرعة تنفيذ النظام.

برنامج التعديل SACUSTED.PRG

يوضـح شكـل ١٢-١٢ برنامج SACUSTED.PRG الذي يستخدم لتعديل بيانـات عميل موجـود بالملف. كما يوضـح شكل ١٣-١٢ العلاقة بين هذا البرنامج والبرامج الأخرى المرتبطة به داخل النظام وعن هذا البرنامج نوضح ما يلي:

- يستخدم نفس الوظيفة الخاصة rec_fnd() التي شرحناها في برنامج SAC-USTED.PRG للتأكد من وجود رقم الحساب بالملف بنفس الطريقة.
- فكرة هذا البرنامج قريبة من فكرة برنامج الحذف SACUSTDE.PRG إلا أننا هنا استخدمنا أمر READ (سطر ٣٤) بعد أوامر GET لنتمكن من تعديل الحقول بدلا من أمر CLEAR GETS في البرنامج السابق.
- بعد تعديل بيانات العميل تظهر رسالة لاعطاء المستخدم الفرصة في تعديل بيانات عميل آخر بناء على رغبته.



شكل ١٣ - ١٢ خريطة برنامج SACUSTED. PRG

قائمة الفواتير Invoices

الاختيار الثاني من القائمة الرئيسية للنظام هو اختيار Invoicesبمعنى الفواتير وهذا الاختيار يظهر قائمة تتيح عمليات إصدار فواتير جديدة (Add) أو تعديل فواتير أصـدرت (Modify) أو إلغـاء فواتير (Delete) ويوضح شكل ١٤-١٢ هذه القائمة. وتشبه فكرتها فكرة قائمة صيانة ملفات العملاء (راجع شكل ٤-١٢) في أنها تستخدم في حالات الاضافة أو التعديل أو الحذف ويشتمل شكل ١٥-١٢ على البرنامج اللازم لاظهار هذه القائمة وتنفيذ اختياراتها. وفكرة هذا البرنامج شبيهة بفكرة البرامج السابقة لاظهار قائمة اختيارات وتلاحظ أنه يستخدم الاجراء inv_fmt وهذا الاجراء خاص بإظهار شكل الفاتورة وأسماء الحقول التي ستظهر والتي ستستخدم في حالات الاضافة أو التعديل أو الحذف. وقد وضعنا هذا الاجراء في آخر البرنامج. وأيضا من المناسب أن توضع الأوامر التي تظهر شكل الفاتورة داخل إجراء مستقل لأنها ستستخدم بواسطة كل البرامج التي تتعامل مع الفواتير.

```
Date: 13/12/1990                                          Time: 09:40:17
                  Add      Modify      Delete      Return

   Account No:                                        Invoice No:
   Sale Date :  / /                                   Salesman   :
   Sold to   :

  Item No    Quan.     Description               Price        Amount

                         Create a new invoice
```

شكل ١٤-١٢ قائمة الفواتير

```
 1 * -----------------------------------------------------------------*
 2 * Program     : SAINV.PRG                                          *
 3 * Author      : Magdi M. Abu Al-Ata                                *
 4 * Date        : October 24, 1990                                   *
 5 * Purpose     : To paint the invoice form                          *
 6 * Copyright   : Magdi M. Abu Al-Ata                                *
 7 * Called from: SAMAIN.PRG                                          *
 8 * -----------------------------------------------------------------*
 9 @1,0 CLEAR TO 24,79
10 DO WHILE .T.
11 |     @  1,14 TO 3, 66  DOUBLE
12 |     @  9,1 TO 23,78  DOUBLE
13 |     @ 11,2 TO 11,77
14 |     DO inv_fmt
15 |     @ 2,20 PROMPT "Add"  MESSAGE "Create a new invoice"
16 |     @ 2,29 PROMPT "Modify"  MESSAGE "Edit an existing invoice"
17 |     @ 2,41 PROMPT "Delete"  MESSAGE "Delete an invoice"
18 |     @ 2,52 PROMPT "Return" MESSAGE "Go to main menu"
19 |     MENU TO choice2
20 |     DO CASE
21 |     CASE choice2 = 0
22 <--------RETURN
23 |     CASE choice2 = 1
24 |     |    DO sainvadd
25 |     CASE choice2 = 2
26 |     |    DO sainvedt
27 |     CASE choice2 = 3
28 |     |    DO sainvdel
29 |     CASE choice2 = 4
30 <--------RETURN
31 |     ENDCASE
32 ENDDO
33
34 ************************
35 *  End of program      *
36 ************************
37
38 *--------------------------------*
39 *   Procedures and Functions     *
40 *   Called only by SAINV.PRG     *
41 *--------------------------------*
42
43 ****************
44 *  Procedure     : inv_fmt
45 *  Purpose       : to drow the invoice form
46 ****************
47 PROCEDURE inv_fmt
48 PUBLIC m_acct, m_acct_no, m_addr, m_company, m_fnam, m_mnam, ;
48  m_lnam, m_city, m_phone, m_inv, m_qty, m_unit_price, m_sal_man, ;
48  m_sal_dat, m_item_no, m_desc, m_ext, m_tax, m_tot
49 * Then it gives those vairables their starting values.
50 STORE CTOD("  /  /  ") TO m_sal_dat
51 STORE 0 TO m_qty, m_unit_price, m_inv, m_ext, m_tot
```

شكل ١٥ ـ ١٢ برنامج SAINV. PRG

```
52 STORE SPACE (12) TO m_mnam
53 STORE SPACE (12) TO m_sal_man
54 STORE SPACE (3)  TO  m_item_no
55 STORE SPACE (4)  TO m_acct, m_acct_no
56 STORE SPACE (14) TO m_phone
57 STORE SPACE (12) TO m_lnam
58 STORE SPACE (17) TO m_city
59 STORE SPACE (12) TO m_fnam
60 STORE SPACE (20) TO m_desc
61 STORE SPACE (20) TO m_company, m_addr
62 @ 5, 6 SAY "Account No:"
63 @ 5,18 SAY m_acct
64 @ 5,51 SAY "Invoice No:"
65 @ 5,62 SAY m_inv PICTURE "@Z 999999"
66 @ 6, 6 SAY "Sale Date :"
67 @ 6,18 SAY m_sal_dat
68 @ 6,51 SAY "Salesman  :"
69 @ 6,62 SAY m_sal_man
70 @ 7, 6 SAY "Sold to   :"
71 @ 7,18 SAY m_fnam
72 @ 7,25 SAY m_mnam
73 @ 7,27 SAY m_lnam
74 @ 8,18 SAY m_company
75 @ 8,39 SAY m_addr
76 @ 8,60 SAY m_city
77 @10, 4 SAY "Item No    Quan.     Description              Price  Amount"
78 @ 12,2 CLEAR TO 22,77  && To erase the items pained
79 RETURN
80
81 ****************************
82 *  End of file SAINV.PRG    *
83 ****************************
```

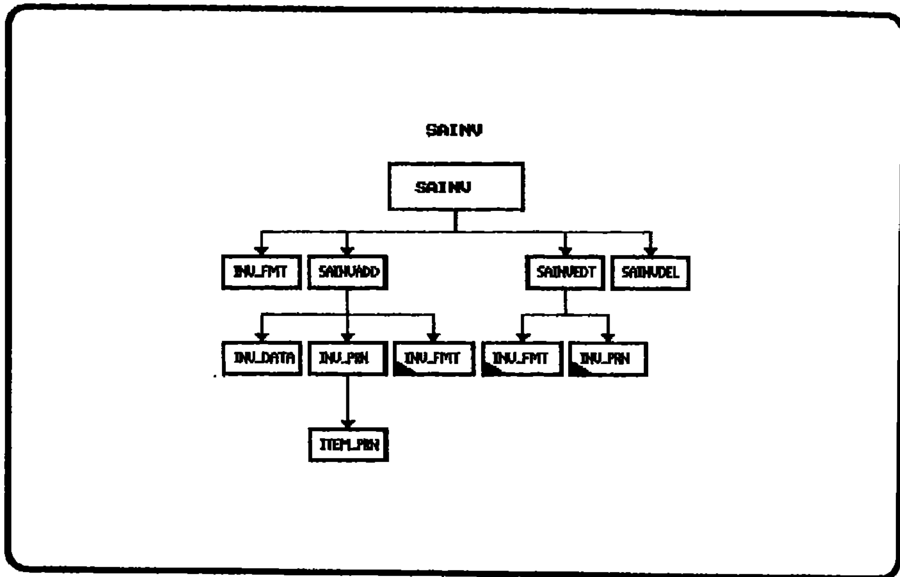
تابع شكل ١٥ - ١٢

ويوضح شكل ١٦-١٢ العلاقة بين برنامج SAINV.PRG والبرامج الأخرى داخل النظام.

ونوضح فيما يلي البرامج الثلاثة التي تستخدم مع الفواتير.

برنامج إصدار فاتورة جديدة SAINVADD.PRG

يوضح شكل ١٧-١٢ برنامج SAINVADD.PRG الذي يستخدم لانشاء فاتورة جديدة كما يوضح شكل ١٨-١٢ العلاقة بين هذا البرنامج والبرامج الأخرى المرتبطة به داخل النظام. وعن هذا البرنامج نوضح ما يلي:



شكل ١٦ ـ ١٢ خريطة برنامج SAINV. PRG

```
 1 * ----------------------------------------------------------------*
 2 * Program     : SAINVADD.PRG                                      *
 3 * Author      : Magdi M. Abu Al-Ata                               *
 4 * Date        : October 24, 1990                                  *
 5 * Purpose     : To create new invoice                             *
 6 * Copyright   : Magdi M. Abu Al-Ata                               *
 7 * Called from: SAINV.PRG                                          *
 8 * Called from: SAMAIN.PRG                                         *
 9 * ----------------------------------------------------------------*
10 @24,0
11 PUBLIC m_dup
12 m_dup = .F.
13 USE custmr INDEX custmr
14 SELECT 0
15 USE invoice INDEX invoice
16 SELECT 0
17 USE inv_item INDEX inv_item
18 DO WHILE .T.
19 |    IF .NOT. FILE("sainvnum.mem")
20 |    |    SAVE ALL LIKE m_inv TO sainvnum
21 |    ENDIF
22 |    RESTORE FROM sainvnum ADDITIVE
23 |    m_inv = m_inv + 1
```

شكل ١٧ ـ ١٢ برنامج SAINVADD. PRG

```
24 |     SAVE ALL LIKE m_inv TO sainvnum
25 |     STORE DATE() to m_sal_dat
26 |     SET ESCAPE ON
27 |     @ 5, 18 GET m_acct PICTURE "9999" VALID val_get()
28 |     READ
29 |     IF LASTKEY() = 27
30 |     |   CLOSE DATABASES
31 |     |   @24,0
32 <--------RETURN
33 |     ENDIF
34 |     SET ESCAPE OFF
35 |     @ 5, 62 SAY m_inv PICTURE '999999'
36 |     @ 6, 18 SAY m_sal_dat
37 |     @ 7, 18 SAY TRIM(custmr->fnam) + " "  + TRIM(custmr->mnam) ;
37 |       + " " + custmr->lnam
38 |     @  8, 18 SAY TRIM(custmr-> company) + "," + TRIM(custmr->addr);
38 |       +"," + TRIM (custmr->city) + "."
39 |     @  6,62 GET m_sal_man
40 |     READ
41 |     REPLACE invoice-> acct_no with m_acct, invoice->inv with m_inv, ;
41 |      invoice->sal_dat with m_sal_dat, invoice->sal_man with m_sal_man
42 |     DO inv_data
43 |     STORE " " TO m_ok
44 |     @ 24,0
45 |     @ 24,25 SAY "Print this invoice? [Y/N]---> " GET m_ok
46 |     READ
47 |     IF UPPER (m_ok) = "Y"
48 |     |   DO inv_prn
49 |     ENDIF
50 |     STOR " " TO m_ok
51 |     @ 24, 0
52 |     @ 24, 20 SAY "Enter another new invoice? (Y/N)---> "GET m_ok
53 |     READ
54 |     @ 12,3 clear to 22,76
55 |     @24,0 CLEAR
56 |     IF UPPER (m_ok) = "Y"
57 |     |   DO inv_fmt
58 |     ELSE
59 |     |   CLOSE  DATABASES
60 <--------RETURN
61 |     ENDIF
62 ENDDO
63
64 ************************
65 *  End of program      *
66 ************************
67
68 *--------------------------------*
69 *   Procedures and Functions     *
70 *   Called  by SAINVADD.PRG      *
71 *--------------------------------*
72
73 ****************
```

تابع شكل ١٧ ـ ١٢

```
74 *  Function       : val_get()
75 *  Purpose        : To validate the invoice number
76 *  Syntax         : val_get()
77 *  Returns        : .T. or .F.
78 *  Example        : IF val_get()
79 ****************
81 FUNCTION val_get
82 SELECT custmr
83 IF m_acct = SPACE (4)
84 |     TONE (349,3)
85 |     TONE (300,5)
86 |     @ 24,0
87 |     @ 24,27 SAY "Press Esc to go back to menu."
88 <----RETURN .F.
89 ENDIF
90 SEEK &m_acct
91 IF EOF()
92 |     TONE(349,3)
93 |     TONE(300,5)
94 |     @ 24,0
95 |     @ 24,20 SAY "Sory. No such account. Try again."
96 <----RETURN .F.
97 ELSE
98 |     SELECT invoice
99 |     APPEND BLANK
100 |     @ 24, 0
101 <----RETURN .T.
102 ENDIF
103
104 ****************
105 *  Procedure      : inv_data
106 *  Purpose        : To process the invoice
107 ****************
108 PROCEDURE inv_data
109 line = 12
110 SELECT inv_item
111 DO WHILE .T.
112 |     @ 24,0
113 |     @ 24,20 SAY "Enter item code or oress ENTER to exit"
114 |     @ line,   4 GET  m_item_no PICTURE '999'
115 |     READ
116 |     IF m_item_no = SPACE (3)
117 |<-------EXIT
118 |     ENDIF
119 |     @ 24,0   && TO erase the message
120 |     @ line,  15 GET  m_qty PICTURE '@z 99'
121 |     @ line,  25 GET  m_desc
122 |     @ line,  51 GET  m_unit_price PICTURE '@Z 9999.99'
123 |     READ
124 |     * Stuff the record with the collected data.
125 |     APPEND BLANK
126 |     REPLACE inv_no with m_inv, item_no with m_item_no  desc with ;
126 |       m_desc, qty with m_qty, unit_price with m_unit_price
```

تابع شكل ١٧ - ١٢

```
127 |     m_ext = m_qty * m_unit_price
128 |     m_tot = m_tot + m_ext
129 |     @ line, 64 SAY m_ext PICTURE '999999.99'
130 |     @ 21, 64 SAY "========="
131 |     @ 22, 40 SAY "Total"
132 |     @ 22, 64 SAY m_tot PICTURE '999999.99'
133 |     IF line <> 20
134 |     |    line = line + 1
135 |     ELSE
136 |     |    SCROLL (12,3,20,76,1)
137 |     ENDIF
138 |     STORE 0 TO m_qty, m_unit_price, m_ext && Reinit the variables.
139 |     STORE SPACE (3) TO m_item_no
140 |     STORE SPACE (20) TO m_desc
141 ENDDO
142 RETURN
143
144 ****************
145 *  Procedure     : inv_prn
146 *  Purpose       : Print reccorded invoice without items
147 ****************
148 PROCEDURE inv_prn
149 SET CURSOR OFF
150 SAVE SCREEN TO prin_inv
151 STORE 0 TO m_tax, m_tot
152 @ 8,15 CLEAR TO 20,60
153 @ 8,15 TO 20,60 DOUBLE
154 @12,20 SAY "Ready to print this invoice. Make"
155 @14,20 SAY "Sure the printer is on and ready."
156 @16,20 SAY "Press any key to begin."
157 INKEY (0)                    && Put a window and pause
158 @ 9,16 CLEAR TO 19,59
159 @14,20 SAY"   Printing Invoice: " + STR(m_inv,6,0)
160 pline = 1
161 page = 1
162 SET DEVICE TO PRINT
163 IF m_dup
164 |    @pline + 2, t_cent("Modified Invoice", 80)  SAY ;
164 |      "Modified Invoice"
165 |    pline = pline + 2
166 ENDIF
167 hd1 = "Your company name"
168 hd2 = "Your address and C.R."
169 @pline + 1, t_cent(hd1,80) SAY hd1
170 @pline + 2, t_cent(hd2,80) SAY hd2
171 pline = pline + 3
172 @pline + 1, 6 SAY "Sold to: " + TRIM(custmr->fnam) + " "  + ;
172   TRIM(custmr->mnam) + " " + custmr->lnam
173 @pline + 1, 51 SAY "Account #: " + custmr->acct
174 @pline + 2, 15 SAY  custmr->company
175 @pline + 2, 51 SAY "Invoice No: " + STR(m_inv,6,0)
176 @pline + 3, 15 SAY TRIM(custmr->addr) + ","
177 @pline + 3, 51 SAY "Sale date: " + DTOC(invoice->sal_dat)
178 @pline + 4, 15 SAY TRIM(custmr->city) + ","
```

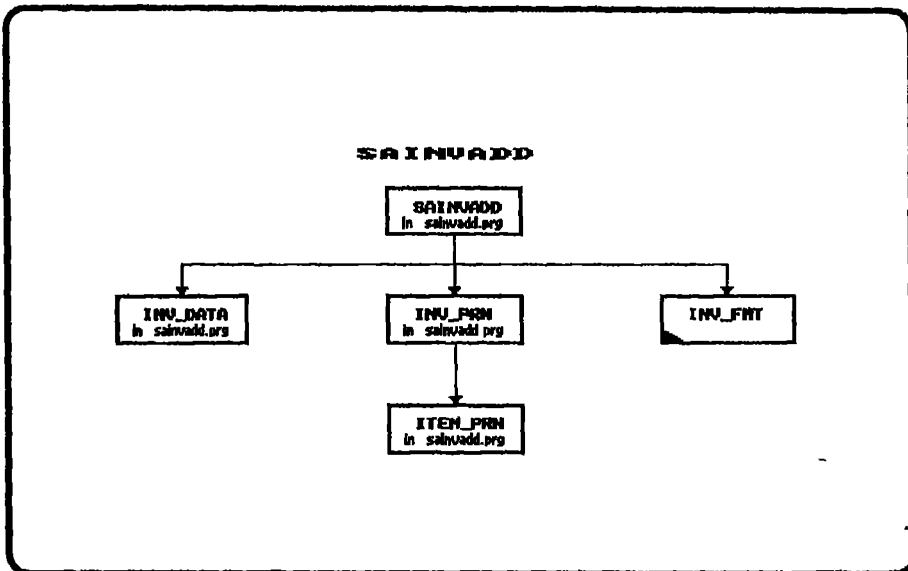
تابع شكل ١٧ - ١٢

```
179 @pline + 4, 51 SAY "Salesman: " + invoice->sal_man
180 @pline + 6, 5 SAY REPLICATE ("-",70)
181 pline = pline + 7
182 @pline + 1, 7 SAY "Item              Item              " + "Unit  Amount   "
183 @pline + 2, 7 SAY "Code    Quan.   Description         " + "Prince        "
184 @pline + 3, 5 SAY REPLICATE ("-",70)
185 pline = pline + 5
186 DO item_prn
187 pline = pline + 2
188 @pline + 3, 49 SAY "    Total: " + STR(m_tot,12,2)
189 EJECT
190 SET DEVICE TO SCREE
191 RESTORE SCREEN FROM prin_inv
192 SET CURSOR ON
193 RETURN
194
195 ****************
196 *  Procedure      : item_prn
197 *  Purpose        : Puts the invoice titems
198 ****************
199 PROCEDURE item_prn
200 SELECT inv_item
201 SET INDEX TO inv_item
202 SEEK m_inv
203 IF EOF()
204 |    @pline + 3, 7 SAY "This invoice has no items to print."
205 <----RETURN
206 ENDIF
207 DO WHILE .T.
208 |    IF pline > 55
209 |    |    EJECT
210 |    |    @pline + 1, 22 SAY hd1
211 |    |    @pline + 2, 22 SAY hd2
212 |    |    @pline + 3, 22 SAY "Page No. " + STR(page,2,0)
213 |    |    *@pline + 5, 5 SAY REPLICATE ("-",70)
214 |    |    pline = pline + 6
215 |    |    @pline + 2, 51 SAY "Invoice No: " + str(m_inv,6,0)
216 |    |    @pline + 3, 51 SAY "Sale Date : " + DTOC(Invoice->sal_dat)
217 |    |    @pline + 4, 51 SAY "Salesman  : " + invoice->sal_man
218 |    |    @pline + 6, 5  SAY REPLICATE ("-",70)
219 |    |    pline = pline + 7
220 |    |    @pline + 1, 7 SAY "Item           Item          " + "Unit  Amount  "
221 |    |    @pline + 2, 7 SAY "Code    Quan.  Description   " + "Price        "
222 |    |    @pline + 3, 5  SAY REPLICATE ("-",70)
223 |    |    pline = pline + 4
224 |    ENDIF
225 |    * Here the item information is printed.
226 |    @pline + 1, 7 SAY item_no
227 |    @pline + 1,15 SAY STR (qty,4,0)
228 |    @pline + 1,25 SAY desc
229 |    @pline + 1,51 SAY STR (unit_price,7,2)
230 |    @pline + 1,64 SAY STR (qty * unit_price,9,2)
231 |    m_tot = m_tot + (qty * unit_price)
```

تابع شكل ١٧ - ١٢

```
232 |     pline = pline + 1
233 |     SKIP
234 |     IF .NOT. EOF() .AND. inv_no = m_inv
235 ^---------LOOP
236 |     ELSE
237 <---------RETURN
238 |     ENDIF
239 ENDDO
240 RETURN
241
242 ******************************
243 *  End of file SAINVADD.PRG *
244 ******************************
```

تابع شكل ١٧ ـ ١٢



شكل ١٨ ـ ١٢ خريطة برنامج SAINVADD. PRG

- سطر ١١، ١٢ لانشـاء حقـل بالذاكرة باسم m_dup ومهمته إخبار برنامج الطباعة (سنشرحه بعد قليل) ما إذا كانت الفاتورة المطبوعة تصدر لأول مرة أم أنها تعديل لفاتورة سابقة. ولذلك وضعنا به القيمة .F. لأن الفاتورة تصدر هنا لأول مرة.

- السطور من ١٣-١٧ لفتح الملفات المطلوبة في ٣ مناطق متجاورة (لاحظ أن أمر SELECT 0 معناه اختيار المنطقة التالية للمنطقة الحالية) والملفات المطلوبة في هذا البرنامج هي :
  ١ ـ ملف CUSTMR.DBF (راجع شكل ٩-١٢)
  ٢ ـ ملف INVOICE.DBF (انظر شكل ١٩-١٢)
  ٣ ـ ملف INV_ITEM.DBF (انظر شكل ٢٠-١٢)
- جملة IF في سطر ١٩ تتأكد من وجود ملف SAINVNUM.MEM على نفس الدليل. ولذلك فإذا كان البرنامج ينفذ لأول مرة فسيتم إنشاء الملف وسيوضع به الرقم صفر. ثم يضاف إليه واحد بعد ذلك. أما إذا كان الملف موجودا أي إذا كان البرنامج نفذ ولو مرة واحدة سابقا فسيتم نقل محتويات الملف إلى الذاكرة (سطر رقم ٢٢) وزيادة واحد لرقم آخر فاتورة سطر رقم ٢٣) ثم حفظ آخر رقم أصدر على الملف مرة ثانية (سطر رقم ٢٤). والهدف من كل ذلك أن يتم ترقيم الفواتير تلقائيا منعا للتلاعب الذي قد يتم من قبل البائع.

```
F1      F2      F3      F4      F5      F6       F7      F8
Help    Open    Create  Save    Browse  Utility  Move    Set

                        Files

Structure of INVOICE.DBF  Field 1

Field Name    Type          Width   Dec
              Character       4
INV           Numeric         6      0
SAL_DAT       Date            8
SAL_MAN       Character      12
```

شكل ١٩ ـ ١٢ مواصفات ملف IN VOICE. DBF

```
F1      F2      F3       F4      F5       F6       F7      F8
Help    Open    Create   Save    Browse   Utility  Move    Set

                         Files

        Structure of  INV_ITEM.DBF Field 1

        Field Name   Type        Width   Dec
        ██████████   Numeric         6     0
        ITEM_NO      Character       3
        DESC         Character      20
        QTY          Numeric         4     0
        UNIT_PRICE   Numeric         7     2
```

شكل ٢٠ - ١٢ مواصفات ملف INV. ITEM. DBF

- السطر رقم ٢٧ يظهر ٤ فراغات لادخال رقم حساب العميل ويستخدم الوظيفة الخاصة val_get() للتأكد أن رقم الحساب سليم. وقد وضعنا هذه الوظيفة في آخر البرنامج. والوظيفة تقوم كذلك بإضافة سجل في نهاية ملف الفواتير إذا وجدت رقم الحساب موجودا بملف العملاء.
- أمر REPLACE في سطر قم ٤١ لنقل البيانات التي دخلت من لوحة المفاتيح إلى ملف الفواتير.
- سطر رقم ٤٢ لاستدعاء إجراء inv_data ووضعنا هذا الاجراء مع الاجراءات الأخرى التي يستخدمها البرنامج وبرامج أخرى في آخر البرنامج (سطر رقم ١٠٨) وهذا الاجراء هام جدا لأنه يقوم بتعبئة الفاتورة بالأصناف المباعة ويبدأ تسجيل أول صنف ابتداء من السطر رقم ١٢ على الشاشة (سطر رقم ١٠٩) ويحتل كل صنف سطرا مستقلا داخل الفاتورة وتتسبب الدوارة في تكرار تسجيل الأصناف وتجميع أسعار المبيعات. تشتمل الدوارة على جملتي IF. الأولى (سطر رقم ١١٦) للخروج من الدوارة إذا ضغط المستخدم مفتاح الادخال بعد آخر صنف تم تسجيله بالفاتورة. والثانية (سطر ١٣٣) لطي الشاشة إذا زادت الأصناف المباعة عن ٩ أصناف وهي المساحة المتوفرة داخل

الشاشة لتسجيل الأصناف المباعة. وتطوى الشاشة بمقدار سطر واحد كلما أدخل صنف جديد بالفاتورة (سطر ١٣٦).

- بعد الانتهاء من تعبئة الفاتورة تظهر رسالة للمستخدم لطباعة الفاتورة فإذا أجاب بنعم استُدعي الاجراء inv_prn لطباعة الفاتورة. والاجراء أيضا موجود في آخـر الـبرنامج وهو لا يختلف كثيرا عن برامج الطباعة العادية وليس فيه مفاهيم غامضة.
- والاجراء بعد أن ينتهي من طباعة بيانات الفاتورة الثابتة والتي لا تتغير من فاتورة لأخرى يستدعي إجراء آخر هو item_prn وهذا الاجراء يتولى طباعة الأصناف. , كل صنف في سطر مستقل من ملف INV_ITEM.DBF
- سواء تمت طباعة الفاتورة أم لم تتم تظهر رسالة أخرى لتكرار إصدار فاتورة أخـرى جديدة (سـطر رقـم ٥٢) فإذا اختـار المستخدم فسينتهي البرنامج. وستمحى المساحة المخصصة لتسجيل الأصناف المباعة من الشاشة.
- أما إذا قرر إصدار فاتورة أخرى فيتم استدعاء الاجراء inv_fmt الموجود في برنامج SAINV.PRG وستتكرر الخطوات التي شرحناها حتى يقرر الخروج من البرنامج.

هـــام

طبقا لمنطق البرنامج سيتم إضافة سجل في نهاية ملف الفواتير (راجع الوظيفة الخاصة val_get()). والمشكلة هنا أن المستخدم إذا قرر إلغاء الفاتورة قبل تعبئة بياناتها فسيظل رقمها موجودا بملف الفواتير. فإذا حذفت هذا الرقم بواسطة أي برنامج أو إجراء فإن هذا الرقم سيكسر تسلسل أرقام الفواتير داخل الملف فمثلا إلغاء الفاتورة رقم ٢٠ سيجعل الرقم ٢١ يظهر بعد الرقم ١٩. فإذا كان تسلسل أرقام الفواتير لا يعني شيئا هاما بالنسبة لك. فهذا المنطق سليم. أما إذا كنت تحرص على هذا التسلسل فيمكنك استخدام منطق آخر يتلخص في تسجيل الفاتورة على ملف مؤقت له نفس مواصفات ملف الفواتير. وبعد الانتهاء من تسجيلها وتقرير أن بياناتها سليمة تنقل الفاتورة برقمها إلى الملف الأصلي وفي النهاية يلغى الملف المؤقت.

برنامج تعديل الفاتورة SAINVEDT.PRG

يوضح شكل ١٢-٢١ برنامج SAINVEDT.PRG الذي يستخدم لتعديل بيانات فاتورة أصدرت. ويشمل هذا التعديل إضافة أصناف جديدة أو حذف أصناف موجودة أو تعديل بيانات صنف مثل السعر أو الكمية. ويتم تعديل بيانات الفاتورة من خلال الشاشة الموجودة في شكل ١٢-١٤. ويوضح شكل ١٢-٢٢ العلاقة بين هذا البرنامج والبرامج الأخرى داخل النظام.

```
 1 * ------------------------------------------------------------------*
 2 * Program     : SAINVEDT.PRG                                        *
 3 * Author      : Magdi M. Abu Al-Ata                                 *
 4 * Date        : October 24, 1990                                    *
 5 * Purpose     : To modify an invoice                                *
 6 * Copyright   : Magdi M. Abu Al-Ata                                 *
 7 * Called from: SAINV.PRG                                            *
 8 * Called from: SAMAIN.PRG                                           *
 9 * ------------------------------------------------------------------*
10 @ 24,0 CLEAR
11 PUBLIC m_dup
12 m_dup = .T.
13 USE custmr
14 SELECT 0
15 USE invoice
16 SELECT 0
17 USE inv_item
18 DO WHILE .T.
19 |     SET ESCAPE ON
20 |     * The VAL_SEEK() function validates the entered invoice number.
21 |     @ 5, 62 GET m_inv PICTURE '999999' VALID val_seek()
22 |     READ
23 |     IF LASTKEY() = 27
24 |     |    CLOSE DATABASES
25 <--------RETURN
26 |     ENDIF
27 |     SET ESCAPE OFF
28 |     SELECT invoice
29 |     @ 5, 18 SAY custmr->acct
30 |     @ 6, 18 SAY invoice->sal_dat
31 |     @ 6, 62 SAY invoice->sal_man
32 |     @ 7, 18 SAY TRIM(custmr->fnam) + " " + TRIM(custmr->mnam) ;
32 |       + " " + custmr->lnam
33 |     @ 8, 18 SAY TRIM(custmr-> company) + "," + ;
33 |       TRIM(custmr->addr) + "," + TRIM (custmr->city) + "."
34 |     * This next statements prepare for the DEBIT().
35 |     DECLARE ar_edit[5]
```

شكل ١٢-٢١ برنامج SAINVEDT. PRG

```
36 |     ar_edit[1] = "item_no"
37 |     ar_edit[2] = "qty"
38 |     ar_edit[3] = "desc"
39 |     ar_edit[4] = "unit_price"
40 |     ar_edit[5] = "qty * unit_price"
41 |     @15,6 CLEAR TO 19,73
42 |     @21,6 CLEAR TO 21,73
43 |     SELECT inv_item
44 |     SET INDEX TO inv_item
45 |     SEEK m_inv
46 |     IF EOF()
47 |     |   TONE(349,3)
48 |     |   TONE(300,5)
49 |     |   @ 24,0
50 |     |   @ 24,20 SAY "ERROR. No invoice items exist."
51 |     |   TONE(300,5)
52 |     |   TONE(349,3)
53 |     |   INKEY(3)
54 |     |   CLOSE DATABASES
55 |     |   @ 24, 0
56 |     |   DO inv_fmt
57 <--------RETURN
58 |     ENDIF
59 |     temp = TRIM(STR(SECONDS(),5,0))  && 5 is all you can get
60 |     COPY TO &temp WHILE inv_no = m_inv
61 |     SELECT 0
62 |     * USE &temp               && Invalid syntax
33 |     USE (temp)
64 |     dummy = ""
65 |     SET DELETED OFF        && Let deleted records to appear
66 |     STORE 0 TO  m_tot
67 |     @ 24,0
68 |     @ 24,27 SAY "Select field and press ENTER."
69 |     DBEDIT (12,3,19,76,ar_edit,"udf_edit",dummy,  dummy, dummy, SPACE(4))
70 |     @ 24,0
71 |     SET DELETED ON        && Disappears deleted records
72 |     USE                   && Close temporary file
73 |     USE inv_item
74 |     DELETE ALL FOR inv_no=m_inv
75 |     APPEND FROM &temp
76 |     temp = TRIM(temp)+".DBF"
77 |     * ERASE &temp          && Invalid syntax
78 |     ERASE (temp)
79 |     DO inv_prn
80 |     CLOSE DATABASES
81 |     DO inv_fmt
82 <----RETURN
83 ENDDO
84
85 ************************
86 *  End of program      *
87 ************************
88
89 *------------------------------*
```

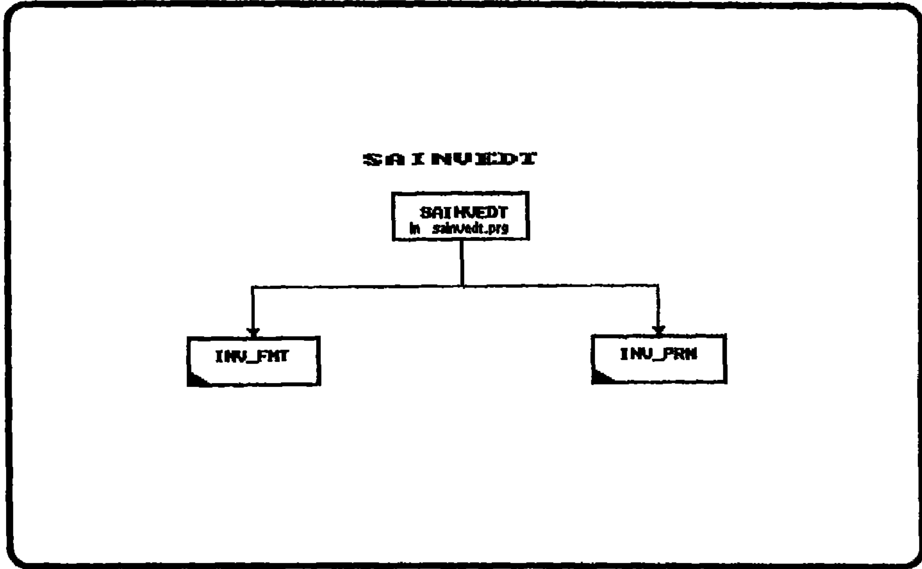
تابع شكل ٢١ - ١٢

```
 90 *   Procedures and Functions     *
 91 *   Called only by SAINVEDT.PRG  *
 92 *-------------------------------*
 93
 94 ****************
 95 *  Function      : val_sek()
 96 *  Purpose       : To validate the entered invoice nounber
 97 *  Returns       : .T. or .F.
 98 *  Syntax        : val_seek()
 99 *  Example       : IF val_seek()
100 ****************
101 FUNCTION val_seek
102 SELECT invoice
103 SET INDEX TO invoice
104 IF m_inv = 0
105 |    TONE(349,3)
106 |    TONE(300,5)
107 |    @ 24,0
108 |    @24, 20 SAY "Press Esc to go back to menu."
109 <----RETURN .F.
110 ENDIF
111 SEEK m_inv
112 IF EOF()
113 |    TONE(349,3)
114 |    TONE(300,5)
115 |    @ 24,0
116 |    @24,20 SAY "Sorry. No such invoice. Try again."
117 <----RETURN .F.
118 ELSE
119 |    STORE acct_no to srch_acct
120 |    SELECT custmr
121 |    SET INDEX TO custmr
122 |    SEEK srch_acct
123 |    * SEEK &srch_acct            && Doesn't work
124 |    IF EOF()
125 |    |   TONE (349,3)
126 |    |   TONE (300,5)
127 |    |   @ 24,0
128 |    |   @ 24,20 SAY "Error..... Customer not on file. "
129 |    |   IF TYPE ("rno") <> "U"
130 <------------RETURN .T.
131 |    |   ENDIF
132 <--------RETURN .F.
133 |    ENDIF
134 <----RETURN .T.
135 ENDIF
136
137 ****************
138 *  Function      : udf_edit()
139 *  Purpose       : To edit invoice items
140 *  Parameters    :
141 *        db_mod  : The current mode
142 *        a_fld   : The field that is highlighted
```

تابع شكل ٢١ - ١٢

```
143 *  Returns       : The current mode and highlighted field
144 *  Example       : DBEDIT (12,3,19,76,ar_edit,"udf_edit")
145 ****************
146 FUNCTION udf_edit
147 PARAMETERS db_mod, a_fld
148 get_fld = ar_edit[a_fld]
149 DO CASE
150 CASE db_mod = 0
151 |      @ 20,2 TO 20,77 DOUBLE
152 |      @ 21,45 TO 22,45 DOUBLE
153 |      @ 21, 7 SAY  "F3: Delete/Undelete item."
154 |      @ 21,56 SAY  "F4: Add an item. "
155 |      @ 22, 7 SAY "Esc: End edit."
156 |      @ 20,33 SAY IIF(DELETED(), "** Deleted **",REPLICATE(CHR(205),13))
157 <----RETURN 1
158 CASE db_mod = 1 .or. db_mod = 2
159 |      ? REPLICATE(CHR(7),2)
160 <----RETURN 1
161 CASE LASTKEY() = -2   && F3 key pressed
162 |      IF DELETED()
163 |      |    RECALL
164 |      ELSE
165 |      |    DELETE
166 |      ENDIF
167 <----RETURN 2
168 CASE LASTKEY() = -3  && F4 key pressed
169 |      APPEND BLANK
170 |      REPLACE inv_no WITH m_inv
171 |      KEYBOARD CHR(1)+CHR(30)
172 <----RETURN 1
173 CASE LASTKEY() = 13 .AND. a_fld<> 5
174 |      SET CURSOR ON
175 |      @ROW(),COL() GET &get_fld
176 |      READ
177 |      cur_rec = RECNO()
178 |      SUM  (qty * unit_price) TO m_tot
179 |      @22,56 SAY "Total:"
180 |      @22,64 SAY m_tot PICTURE '9999999.99'
181 |      GO cur_rec
182 |      SET CURSOR OFF
183 <----RETURN 2
184 CASE LASTKEY() = 27    && Esc key pressed
185 <----RETURN 0
186 OTHERWISE
187 <----RETURN 1
188 ENDCASE
189
190 ****************************
191 *  End of file SAINVEDT.PRG *
192 ****************************
```

تابع شكل ٢١ ـ ١٢



شكل ٢٢ ـ ١٢ خريطة برنامج SAINVEDT. PRG

وعن هذا البرنامج نوضح ما يلي:

- في سطر رقم ١٢ يشتمل حقل الذاكرة على القيمة .T.لنوضح لبرنامج الطباعة أن هذه الفاتورة معدلة وليست جديدة.
- يستخدم البرنامج الوظيفة الخاصة ()val_seek ليتأكد من وجود رقم الفاتورة بالملف. وهذه الوظيفة تستخدم بنفس الطريقة مع برنامج حذف الفاتورة الذي سيرد بعد ذلك وهذه الوظيفة موجودة في آخر البرنامج. وهي تتأكد أولا من وجود رقم الفاتورة في ملف الفواتير. فإذا وجدته بحثت في ملف العملاء عن رقم الحساب المطابق للرقم الموجود بملف الفواتير.
- الأوامر الموجودة ابتداء من سطر ٣٥ تعتبر من أعقد أجزاء النظام وذلك لأنها تستخدم الوظيفة ()DBEDIT وهي أيضا من أعقد الوظائف لاضافة أو حذف أو تعديل أصناف داخل الفاتورة إلا أنها ذات فائدة كبيرة بالنسبة لك في استخدام نفس المفهوم في برامج مشابهة.

وتبدأ هذه الأوامر بإنشاء المصفوفة ar_fld لتشتمل على الحقول التي من المحتمل تعديلها بعد ذلك (سطر ٣٥). ثم تعبئة عناصر المصفوفة بأسماء

الحقول (سطر من ٣٦ إلى ٤٠).

- سطر ٤٥ يضع المؤشر عند أول صنف داخل ملف INV_ITEM.DBF من أصناف الفاتورة المطلوبة فإذا لم يجد البرنامج رقم الفاتورة في ملف الأصناف تظهر رسالة خطأ (سطر ٥٠) ويسمع صوت الجرس.
- سطر ٥٣ لايقاف البرنامج لمدة ٣ ثوان قبل غلق الملفات واستدعاء الاجراء inv_fmt لرسم شكل الفاتورة.
- الأوامر الباقية من البرنامج خاصة بتعديل سجلات الفاتورة فإذا وضع المؤشر عند أول صنف داخل ملف INV_ITEM.DBF (أي إذا وجدت الفاتورة المطلوب تعديلها) فسيتم تعديل أصنافها باستخدام الوظيفة DBEDIT(). وتلاحظ أن المنطق المستخدم لتعديل أصناف الفاتورة يتلخص في نسخ الأصناف المطلوب تعديلها على ملف مؤقت واخترنا له أيضا اسما يساوي عدد الثواني المقابلة للوقت الحالي داخل الحاسب. والسبب في ذلك أن نفس النظام الذي بين أيدينا سيتم تطويره قبل نهاية هذا الفصل للعمل داخل شبكة اتصالات وفي شبكة الاتصالات يحتمل أن يظهر نفس السجل أمام أكثر من مستفيد لذلك اخترنا أن يتم تعديل سجلات الفاتورة على ملف مستقل. أما في حالة إعداد النظام لخدمة مستفيد واحد فيمكن اتباع طريقة أسهل تتلخص في استخدام أمر SET FILTER لاستخراج سجلات الفاتورة فقط ثم تعديلها على نفس الملف وإليك توضيح ذلك من خلال أوامر البرنامج.
- سطر ٥٩ يحول وقت الحاسب إلى عدد من الثواني ويضعها في حقل بالذاكرة اسمه temp. والسبب في اختيار هذا الاسم أن يكون اسم الملف مميزا إذا أردا أكثر من مستفيد تعديل محتويات الفاتورة لأن الرقم سيختلف من ثانية لأخرى.
- سطر رقم ٦٠ ينسخ سجلات هذه الفواتير إلى ملف جديد باسم مطابق للرقم الناتج من الأمر السابق. واستخدمنا الاختيار WHILE لأنه أسرع إلا أنه يتطلب أن يكون الملف مفهرسا.
- سطر ٦٣ يفتح الملف المؤقت.
- سطر ٦٥ لكي تظهر السجلات المعلمة للحذف لنتمكن من استرجاعها

حسب منطق الوظيفة المستخدمة للتعديل.

- سطر ٦٨ لاظهار رسالة في آخر سطر لتوضيح كيفية التعامل مع شاشة التعديل.
- سطر ٦٩ يسمح بإجراء التعديلات داخل مستطيل. واستخدمنا معه وظيفة خاصة باسم udf_edit(). تجدها في آخر البرنامج.

ملاحظة: ننصحك بمراجعة مرجع الوظائف في الباب الرابع للحصول على معلومات أكثر عن الوظيفة DBEDIT() وكيفية استخدام وظيفة خاصة معها.

- سطر ٧١ عكس سطر ٦٥ لاخفاء السجلات المعلمة للحذف حتى لا تظهر مع باقي الأوامر.
- بعد إجراء التعديلات على الملف المؤقت تم إغلاقه (سطر ٧٢) ثم فتحنا ملف الأصناف وحذفنا كل السجلات التي تخص الفاتورة التي بين أيدينا قبل التعديل (سطر ٧٤) ثم نسخنا السجلات التي تعدلت في الملف المؤقت (سواء أضيف إليها أو حذف منها أو تعدلت بياناتها) نسخناها في ملف الأصناف بحالتها بعد التعديل (سطر ٧٥) ثم حذفنا الملف المؤقت لأنه لم تعد هناك حاجة إليه (سطر ٧٨).
- سطر ٧٩ لاستدعاء الاجراء inv_prn لطباعة الفاتورة وهو نفس الاجراء الذي يستخدمه البرنامج السابق.

ملاحظة: لاحظ أن تعديل أحد الحقول في الفاتورة التي ستظهر داخل المستطيل يتم بنقل الشريط المضاء إلى هذا الحقل ثم ضغط مفتاح الادخال. عندئذ سيظهر المؤشر للدلالة على أن التعديل أصبح متاحا. ويجب ضغط مفتاح الادخال مرة ثانية بعد إجراء التعديل المطلوب للخروج من حالة التعديل وإخفاء المؤشر.

برنامج حذف الفاتورة *SAINVDEL.PRG*

يوضح شكل ٢٣-١٢ برنامج SAINVDEL.PRG الذي يستخدم لحذف فاتورة من الملف ويستخدم نفس الشاشة الموجودة في شكل ١٤-١٢.

```
1 * ---------------------------------------------------------------------*
2 * Program    : SAINVDEL.PRG                                             *
3 * Author     : Magdi M. Abu Al-Ata                                      *
4 * Date       : October 24, 1990                                         *
5 * Purpose    : To delete an existing invoice                            *
6 * Copyright  : Magdi M. Abu Al-Ata                                      *
7 * Called from: SAINV.PRG                                                *
8 * Called from: SAINVDEL.PRG                                             *
9 * ---------------------------------------------------------------------*
10 @ 24,0 CLEAR
11 USE custmr INDEX custmr
12 SELECT 0
13 USE invoice INDEX invoice
14 SELECT 0
15 USE inv_item INDEX inv_item
16 rno = RECNO()
17 SET ESCAPE ON
18 @ 5, 62 GET m_inv PICTURE '999999' VALID val_seek()
19 READ
20 IF LASTKEY() = 27
21 |    CLOSE DATABASES
22 <----RETURN
23 ENDIF
24 SET ESCAPE OFF
25 @ 5, 18 SAY custmr->acct
26 @ 6, 18 SAY invoice->sal_dat
27 @ 6,62 SAY invoice->sal_man
28 @ 7, 18 SAY TRIM(custmr->fnam) + " "  + TRIM(custmr->mnam) + " " + custmr->lnam
29 @  8, 18 SAY TRIM(custmr-> company) + "," + TRIM(custmr->addr) ;
29    + "," + TRIM (custmr->city) + "."
30 SELECT inv_item
31 SEEK m_inv
32 rno = RECNO()
33 IF EOF()
34 |     TONE (349,3)
35 |     TONE(300,5)
36 |     @ 24, 0
37 |     @ 24,27 SAY "ERROR. No invoice items exist.    "
38 |     TONE(300,5)
39 |     TONE(349,3)
40 ENDIF
41 line = 12
42 FOR i = 1 TO 10
43 @ line,  4 SAY item_no
44 @ line, 15 SAY STR(QTY,4,0)
45 @ LINE, 25 SAY desc
46 @ line, 51 SAY STR(unit_price,7,2)
47 SKIP
48 IF inv_no = m_inv
49 |    line = line + 1
50 ELSE
51 |    EXIT
52 ENDIF
```

شكل ٢٣ - ١٢ برنامج SAINVDEL. PRG

```
53 NEXT
54 STORE ' ' TO m_ok
55 @ 24,0
56 @ 24, 22 SAY "Delete this invoice? (Y/N)--> "GET m_ok
57 READ
58 @ 24,0
59 IF UPPER (m_ok) = "Y"
60 |    DELETE ALL FOR inv_no = m_inv
61 |    SELECT  invoice
62 |    DELETE
63 ENDIF
64 CLOSE DATABASES
65 RETURN
66
67 *****************************
68 *  End of file SAINVDEL.PRG *
69 *****************************
```

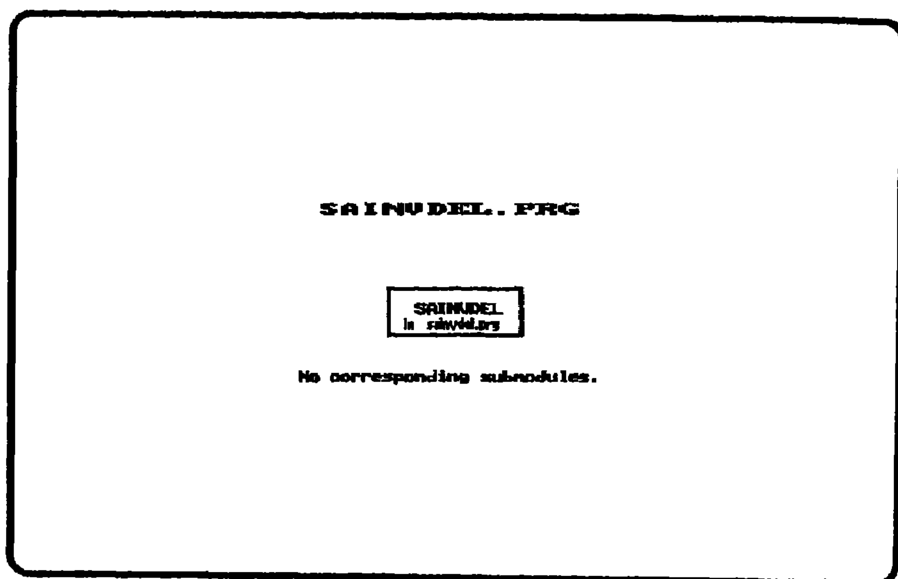
تابع شكل ٢٣ - ١٢

ويوضح شكل ٢٤-١٢ العلاقة بين هذا البرنامج والبرامج الأخرى داخل النظام.

وعن هذا البرنامج نوضح ما يلي:

- يعتبر هذا البرنامج أسهل البرامج الثلاثة التي تتعامل مع الفواتير وهو يستخدم الوظيفة val_seek() بنفس الطريقة التي يستخدمها برنامج التعديل.
- إذا لم يجد الفاتورة المطلوبة يظهر رسالة ويطلق صوت الجرس جملة (IF EOF() في سطر ٣٣.
- إذا وجدها يستخدم أمر FOR...NEXT لاظهار أصنافها داخل مستطيل للعشرة أصناف الأولى (السطور ٤٢-٥٣).

- بعد إظهار العشرة الأصناف الأولى من الفاتورة المطلوبة تظهر رسالة للتأكيد قبل حذف الفاتورة (سطر ٥٦). فإذا أجاب المستخدم بنعم فيتم حذف كل الأصناف من ملف الأصناف INV_ITEM.DBF كما يتم حذف الفاتورة من ملف الفواتير INVOICE.DBF.



شكل ٢٤ ـ ١٢ خريطة برنامج SAINVDEL. PRG

قائمة التقارير (Reports)

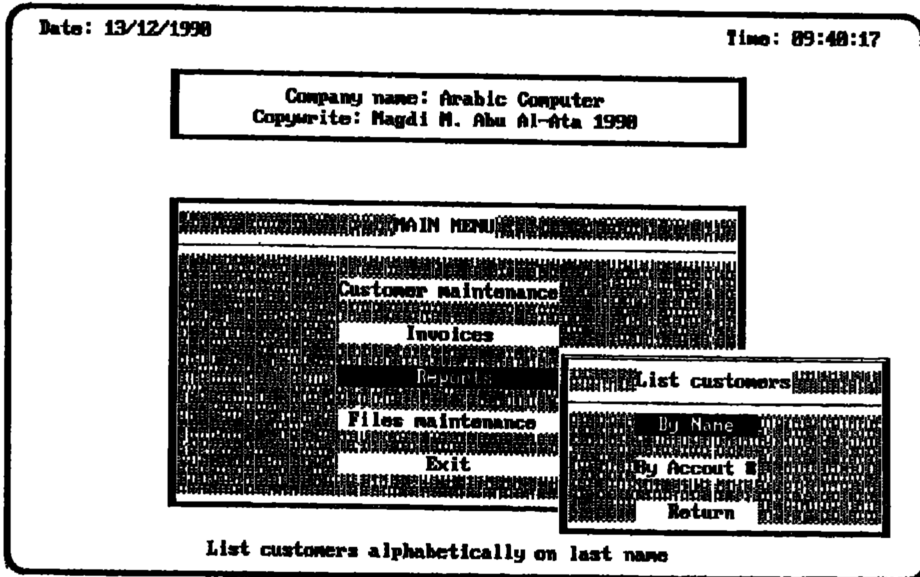
الاختيار الثالث من القائمة الرئيسية للنظام هو الاختيار Reports وهي يظهر قائمة تابعة للتقارير ويوضح شكل ٢٥-١٢ هذه القائمة.

ورغم بساطة هذه القائمة إلا أننا أردنا من خلالها توضيح فكرة إظهار قائمة تابعة لقائمة أخرى على نفس الشاشة وكذلك فكرة إظهار التقرير على الشاشة أو الطابعة ويمكنك إضافة تقارير أخرى إلى هذه القائمة مثل تقارير عن مبيعات فترة زمنية معينة. أو مبيعات كل بائع على حده لمعرفة نسبة العمولة التي تخصه... وهكذا.

ويوضح شكل ٢٦-١٢ البرنامج اللازم لاظهار قائمة التقارير وتنفيذ اختياراتها كما يوضح شكل ٢٧-١٢ العلاقة بين هذا البرنامج والبرنامج الأخرى داخل النظام.

وعن هذا البرنامج نوضح ما يلي:

- يظهر هذا التقرير بيانات العملاء مرتين حسب أبجديات الأسماء أو حسب رقم الحساب. ولذلك فالاجراء في الحالتين واحد والاختلاف في طريقة ترتيب الملف (اختيار الفهرس).



شكل ٢٥ - ١٢ قائمة التقارير

```
1 * ---------------------------------------------------------------*
2 * Program      : SAREP.PRG                                       *
3 * Author       : Magdi M. Abu Al-Ata                             *
4 * Date         : October 24, 1990                                *
5 * Purpose      : To list customers by name or account noo.       *
6 * Copyright    : Magdi M. Abu Al-Ata                             *
7 * Called from: SAMAIN.PRG                                        *
8 * ---------------------------------------------------------------*
9 DO WHILE .T.
10 |     @15, 50 CLEAR TO 23,79
11 |     @15, 50, 23, 79 BOX DUBL
12 |     @16,57 SAY "List customers"
13 |     @17, 51 TO 17,78
14 |     * Create a sub menu.
15 |     @18,57 PROMPT "  By Name  "  MESSAGE "List customers alphabetically on last name"
16 |     @20,57 PROMPT "By Accout #"  MESSAGE "List customers by account number"
17 |     @22,57 PROMPT "   Return  "  MESSAGE "Return to the previous menu"
18 |     MENU TO choice5
19 |     USE custmr
20 |     DO CASE
21 |     CASE choice5 = 0
22 <--------RETURN
23 |     CASE choice5 = 1
24 |     |    SET INDEX TO name
25 |     |    DO rep_cust WITH "Customers quick report by name"
26 |     CASE choice5 = 2
27 |     |    SET INDEX TO custmr
28 |     |    DO rep_cust WITH "Customers quick report by account no."
29 |     CASE choice5 = 3
30 <--------RETURN
31 |     ENDCASE
32 ENDDO
33
34 ************************
35 *  End of program      *
36 ************************
37
38 *-------------------------------*
39 *   Procedures and Functions    *
40 *   Called only by SAREP.PRG    *
41 *-------------------------------*
42
43 ****************
44 *  Procedure    : rep_cust
45 *  Purpose      : prints customers on terminal or printer
46 ****************
47 PROCEDURE rep_cust
48 PARAMETERS pghd
49 PRIVATE cho,pr,pgno,pghd,pghd1,goon
50 STORE 1 TO pgno
51 STORE REPLICATE(CHR(178), LEN(pghd)) TO pghd1
52 STORE '?' TO goon
53 STORE " " TO cho
54 SAVE SCREEN TO rep_scr
```

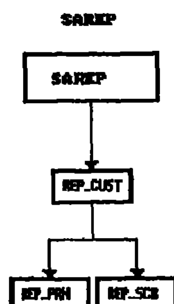
شكل ٢٦ - ١٢ برنامج SAREP. PRG

```
55 CLEAR
56 @ 02,13 SAY "Direct output to the Printer/Terminal? (P/T)"  GET cho
57 READ
58 DO CASE
59 CASE UPPER(cho) = "P"
60 |     DO rep_prn
61 CASE UPPER(cho) = "T"
62 |     DO rep_scr
63 ENDCASE
64 RESTORE SCREEN FROM rep_scr
65
66 ****************
67 *  Procedure    : rep_prn
68 *  Purpose      : prints customers on printer
69 ****************
70 PROCEDURE rep_prn
71 WAIT " The report will direct to the printer. Ready? [Y/N] " TO PR
72 IF UPPER(PR)="Y"
73 |      ? " Please wait ... Report printing "
74 |      SET DEVICE TO PRINT
75 |      SET CONSOLE OFF
76 |      DO WHILE .NOT. EOF()
77 |      |    @ 2,5 SAY "Page No: "+STR(PAGE,3)
78 |      |    @ 3,5 SAY "Date: " + dtoc(date())
79 |      |    @ 5,t_cent(pghd,80)  SAY  pghd
80 |      |    @ 6,t_cent(pghd1,80) SAY  pghd1
81 |      |    @ 8,1 SAY "Account #              Customer name  "
82 |      |    list next 50 acct+space(7)+trim(fnam)+trim(mnam)+lnam
83 |      |    pgno = pgno+1
84 |      |    EJECT
85 |      ENDDO
86 |      SET DEVICE TO SCREEN
87 |      SET CONSOLE ON
88 |      ? " Report finished ... Remove paper "
89 ELSE
90 <----RETURN
91 ENDIF
92
93 ****************
94 *  Procedure    : rep_scr
95 *  Purpose      : prints customers on screen
96 ****************
97 PROCEDURE rep_scr
98 @ 1,0  CLEA TO 21,78
99 @ 2,1 SAY "Page No. " + STR(pgno,3,0)
100 @ 2,60 SAY "Date: " +dtoc (date())
101 @ 3,t_cent(pghd,80) SAY pghd
102 @ 4,t_cent(pghd1,80) SAY pghd1
103 @ 6,  1 SAY REPLI("-",78)
104 ?"Account #               Customer name  "
105 @ 8,  1 SAY REPLI("-",76)
106 DO WHILE .NOT. EOF()
107 |     @ 2,11 SAY STR(pgno,3,0)
108 |     @ 9,1 CLEAR to 21,79
```

تابع شكل ٢٦-١٢

```
109 |     @ 09, 1
110 |     LIST OFF NEXT 10 ACCT+SPACE(7)+TRIM(FNAM)+" "+TRIM(MNAM)+" "+ LNAM
111 |     IF EOF()
112 |     |    @ row() + 2,1 SAY " *** End of data *** Press any key "
113 |     |    INKEY(3)
114 <--------RETURN
115 |     ENDIF
116 |     pgno = pgno+1
117 |     @ 23,1 SAY "Continue ...?(Y/N)" GET goon
118 |     READ
119 |     IF UPPER(goon) = "N"
120 |<-------EXIT
121 |     ENDIF
122 ENDDO
123
124 ****************************
125 *  End of file SAREP.PRG   *
126 ****************************
```

تابع شكل ٢٦-١٢



شكل ٢٧ - ١٢ خريطة برنامج SAREP. PRG

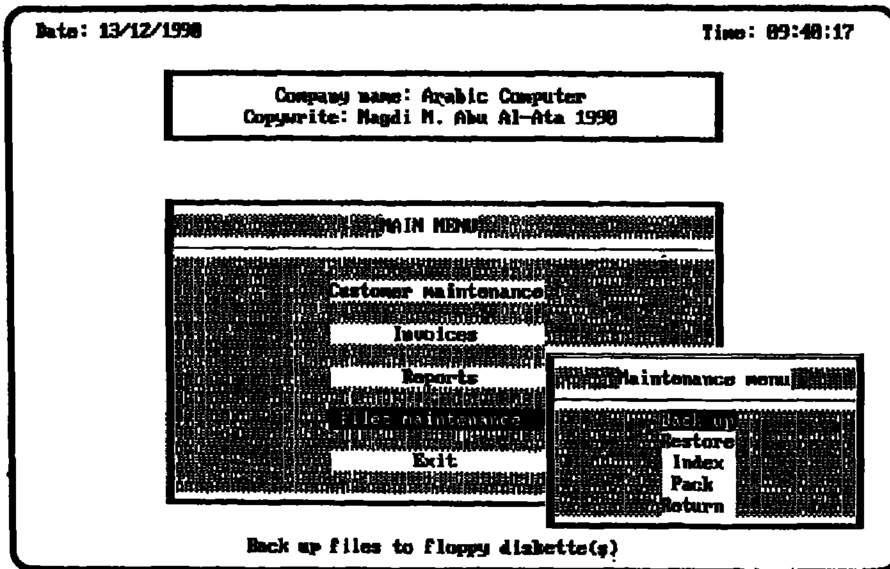
- يبـدأ التقرير بإظهار رسالة لتوجيه التقرير إلى الشاشة أو الطابعة وبناء على اختيار المستخـدم ينفذ إجراء الطباعة المناسب rep_prn أو rep_scr وكلاهما موجود في آخر البرنامج .
- تتلخص فكرة إجراء الطباعة على الطباعة في إلغاء عمل الشاشة أثناء الطباعة واختيار الـطابعة (سطر ٧٤، ٧٥) ثم طباعة كل ٥٠ سطر في صفحة بعد طباعة عنوان الصفحة وبعد الانتهاء من الطباعة تعود كل من الطابعة والشاشة إلى حالتها السابقة (سطر ٨٦، ٨٧) .
- تتلخص فكرة إظهار التقرير على الشاشة في طباعة عنوان الشاشة ثم طباعة كل ١٠ سجلات في شاشة وتوقف الشاشة مؤقتا لمتاعبة التقرير وتوجيه سؤال بالاستمرار في العرض أو الانتهاء. فإذا قرر المستخدم إنهاء العرض توقف البرنامج أو إذا انتهت البيانات توقف البرنامج .

قائمة صيانة الملفات Maintenance

الاختيار الأخير في القائمة الرئيسية للنظام هو قائمة صيانة ملفات النظام. ويوضح شكل ٢٨-١٢ الاختيارات التي تشتمل عليها قائمة صيانة الملفات وهي:

- عمل نسخ احتياطية من الملفات Backup
- استرجاع الملفات المحفوظة Restore
- فهرسة أو إعادة فهرسة الملفات Index
- تنظيف الملفات وحذف السجلات الغير مطلوبة Pack

وهي كما ترى وظائف ضرورة ومشتركة مع كل أنظمة إدارة قواعد البيانات.



شكل ٢٨ - ١٢ قائمة صيانة الملفات

ويشتمل شكل ٢٩-١٢ على برنامج SAUTIL.PRG اللازم لاظهار القائمة وتنفيذ اختياراتها كما يوضح شكل ٣٠-١٢ العلاقة بين هذا البرنامج وغيره من البرامج داخل النظام.

```
 1 * ------------------------------------------------------------*
 2 * Program    : SAUTIL.PRG                                     *
 3 * Author     : Magdi M. Abu Al-Ata                            *
 4 * Date       : October 24, 1990                               *
 5 * Purpose    : Packs up, Restore, Index and Packs data files  *
 6 * Copyright  : Magdi M. Abu Al-Ata                            *
 7 * Called from: SAMAIN.PRG                                     *
 8 * ------------------------------------------------------------*
 9 DO WHILE .T.
10 |      @15, 50 CLEAR TO 23,79
11 |      @15, 50, 23, 79 BOX dubl
12 |      @16,57 SAY "Maintenance menu"
13 |      @17, 51 TO 17,78
14 |      * Create a mini menu.
15 |      @18,61 PROMPT "Back up"  MESSAGE "Back up files to floppy diskette(s)"
16 |      @19,61 PROMPT "Restore"  MESSAGE "Restore the data files from floppy diskettte(s)"
17 |      @20,61 PROMPT " Index "  MESSAGE "Rebuild the index files"
18 |      @21,61 PROMPT " Pack  "  MESSAGE "Remove records marked for deletion"
19 |      @22,61 PROMPT "Return "  MESSAGE "Return to the previous menu"
20 |      MENU TO choice4
21 |      msg = "This option will take between 1 and 20 minutes"
22 |      yn = " "
23 |      DO CASE
24 |      CASE choice4 = 0
25 <--------RETURN
26 |      CASE choice4 = 1
27 |     |    DO saback
28 |      CASE choice4 = 2
29 |     |    DO sarest
30 |      CASE choice4 = 3
31 |     |    DO saindex
32 |      CASE choice4 = 4
33 |     |    DO sapack
34 |      CASE choice4 = 5
35 <--------RETURN
36 |      ENDCASE
37 ENDDO
38
39 ************************
40 *  End of program      *
41 ************************
42
43 *------------------------------*
44 *   Procedures and Function    *
45 *   Called by SAUTIL.PRG       *
46 *------------------------------*
47
48 ****************
49 *  Procedure    : saback
50 *  Purpose      : Bcck up data files to floppy diskettes
51 ****************
52 PROCEDURE saback
53 SAVE SCREEN
54 CLEAR
```

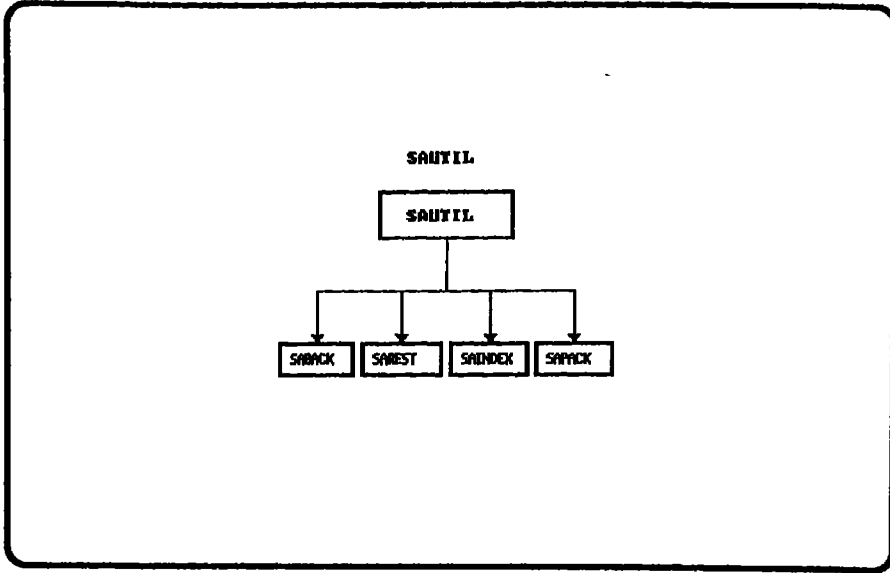
شكل ٢٩ - ١٢ برنامج SAUTIL. PRG

```
55 @ 1,2 SAY MSG
56 @ 2,2 SAY  "Do you have formated diskette(s) for back up? [Y/N]" GET yn
57 READ
58 IF UPPER(yn) = "Y"
59 |    source = "C:"
60 |    target = "A:"                  && Target drive for backup
61 |    files1 = source+"CUSTMR.*"
62 |    files2 = source+"INV*.*"       && To include all invoice and
63 |    && inv item files
64 |    RUN BACKUP &files1 &target
65 |    RUN BACKUP &files2 &target/A   && /A Append those files to existing
66 |    && files on the same drive
67 |    ??CHR(7)
68 |    CLEAR
69 |    WAIT  "End backing up ...Remove your diskettes and press any key to continue"
70 ENDIF   (yn = "Y")
71 RESTORE SCREEN
72 RETURN
73
74 ****************
75 *  Procedure    : SAREST
76 *  Purpose      : Restore data files from floppy diskettes
77 ****************
78 PROCEDURE sarest
79 SAVE SCREEN
80 CLEAR
81 @ 1,2 SAY MSG
82 @ 2,2 SAY  "Do you have your diskette(s) for restore? [Y/N]" GET yn PICT "!"
83 READ
84 IF yn = "Y"
85 |    source = "A:"               && Source drive for restore
86 |    target = "C:"               && Target drive to accept files
87 |    files1 = target+"*.*"       && All files in drive A: will be
88 |    && .DBF, .DBT and .NTX
89 |    RUN RESTORE &source &files1
90 ENDIF (yn = "Y")
91 RESTORE SCREEN
92 RETURN
93
94 ****************
95 *  Procedure    : SAINDEX
96 *  Purpose      : To create new indexes or rebuild existing ones
97 ****************
98 PROCEDURE saindex
99 SAVE SCREEN
00 USE custmr
01 @24,0
02 @24,20 SAY "Please wait...Rebuilding CUSTMR.NTX"
03 IF .NOT. FILE("custmr.ntx")
04 |    INDEX ON acct TO custmr
05 ELSE
06 |    SET INDEX TO custmr
07 |    REINDEX
```

تابع شكل ٢٩-١٢

```
08 ENDIF
09 @24,20 SAY "Please wait...Rebuilding NAME.NTX"
10 IF .NOT. FILE("name.ntx")
11 |     INDEX ON fnam+mnam+lnam TO name
12 ELSE
13 |     SET INDEX TO name
14 |     REINDEX
15 ENDIF
16 USE invoice
17 @24,0
18 @24,20 SAY "Please wait...Rebuilding INVOICE.NTX"
19 IF .NOT. FILE("invoice.ntx")
20 |     INDEX ON inv TO invoice
21 ELSE
22 |     SET INDEX TO invoice
23 |     REINDEX
24 ENDIF
25 USE inv_item
26 @ 24,0
27 @24,20 SAY "Please wait...Rebuilding INV_ITEM.NTX"
28 IF .NOT. FILE("inv_item.ntx")
29 |     INDEX ON inv_no TO inv_item
30 ELSE
31 |     SET INDEX TO inv_item
32 |     REINDEX
33 ENDIF
34 RESTORE SCREEN
35 RETURN
36
37 ****************
38 *  Procedure     : SAPACK
39 *  Purpose       : To delete records marked for deletion
40 ****************
41 PROCEDURE sapack
42 SAVE SCREEN
43 @ 24,0
44 @24,22 SAY "Please wait...Pcking records marked for deletion"
45 USE custmr
46 PACK
47 USE invoice
48 PACK
49 USE inv_item
50 PACK
51 CLEAR
52 CLOSE DATABASES
53 RESTORE SCREEN
54 RETURN
55
56 ****************************
57 *  End of file SAUTIL.PRG   *
58 ****************************
```

تابع شكل ٢٩-١٢



شكل ٣٠ ـ ١٢ خريطة برنامج SAUTIL. PRG

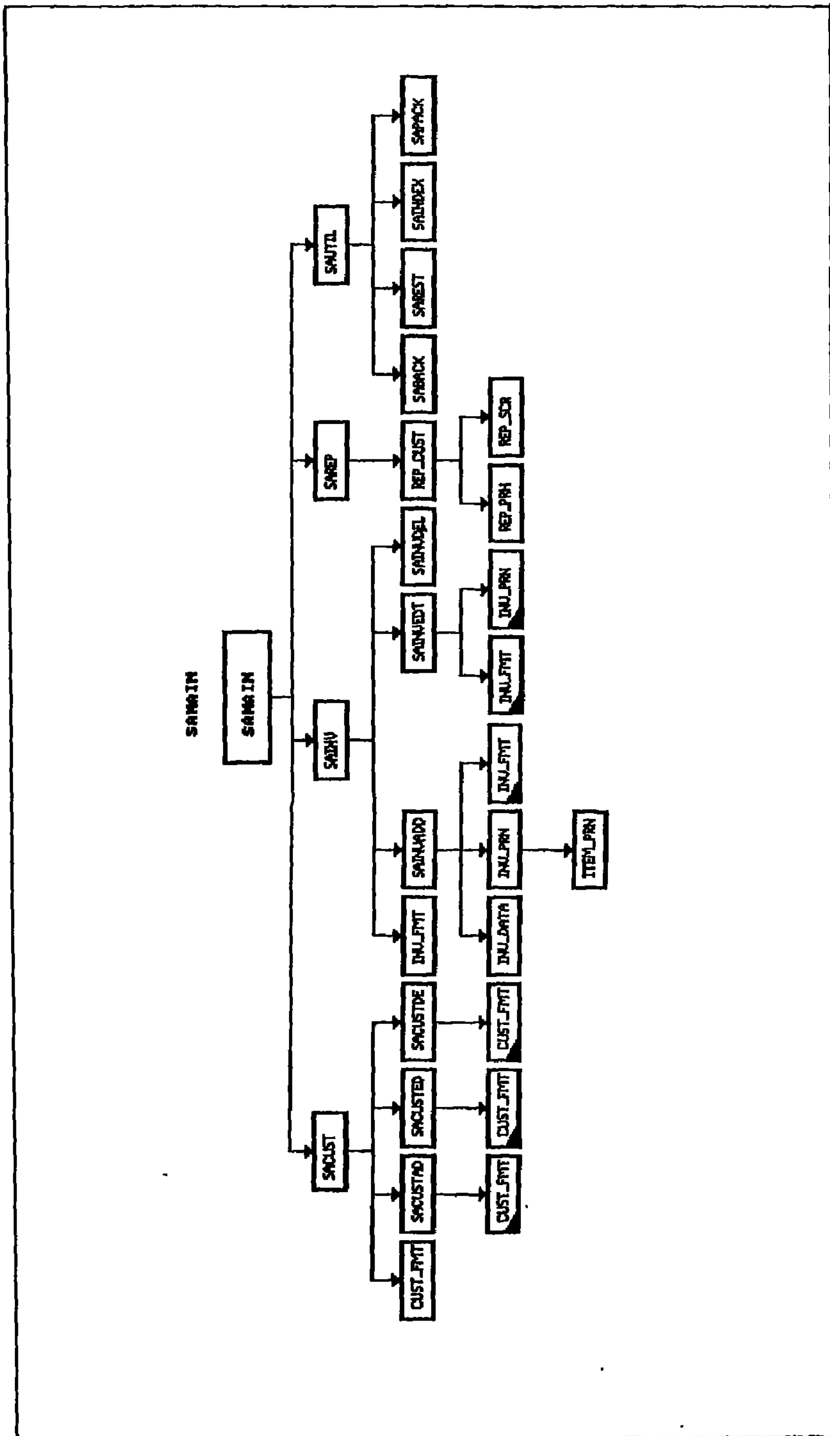
وعن هذا البرنامج نوضح ما يلي:

يشتمل البرنامج على أربعة إجراءات يختص كل إجراء بوظيفة من الوظائف الأربعة.

ـ الاجراء الأول SABACK يستخدم أمر RUN لاستدعاء ملف BACKUP الموجود في نظام التشغيل DOS لعمل نسخ احتياطية للملفات بينما يستخدم الاجراء الثاني SAREST أمر RESTORE لاسترجاع النسخ الاحتياطية.

الاجراء الثالث SAINDEX يقوم بإنشاء ملفات الفهرسة لمرة واحدة فقط ففي كل مرة يتم استدعاؤه للتنفيذ يتأكد أولا من وجود ملف الفهرس (سطر ١٠٣) فإذا لم يجده أنشأه بأمر INDEX ON (سطر ١٠٤) أما إذا وجده فيقوم بإعادة الفهرسة (سطر ١٠٦، ١٠٧) والاجراء الرابع يحذف السجلات المعلمة للحذف والغير مطلوبة من جميع الملفات (سطر ١٤١).

انظر شكل ٣١-١٢ لتتعرف على الخريطة الكاملة لنظام إدارة قواعد البيانات الذي بين أيدينا وهو عبارة عن تجميع لكل الخرائط التي أوردناها لبرامج النظام.



شكل ١٢-٣١ الخريطة الشاملة للنظام

استخدام النظام داخل شبكة الاتصالات

الفـرق بين النظام الذي يعد لخدمة مستفيد واحد أو ذلك الذي يعد لخدمة مجموعة مستفيدين داخل شبكة اتصالات أن النظام الأول يكفيه أن يفتح المستفيد الملف أو السجل ثم يتعامل معه بكافة صور التعديل من غير خوف أن يتأثر الملف أو السجل بتعديلات أخرى لأن المستفيد هنا يضمن أنه هو الوحيد الذي يتعامل مع النظام.

أما النظام الثاني الذي يستخدم مع شبكة اتصالات فيجب أن يغلق المستفيد الملف أو السجل الذي ينوي تعديل أو تغيير محتوياته ليضمن أن أحدا غيره لن يعدل نفس الملف أو السجل ولن يحصل تضارب أو تكرار في التعديل.

ولذلك لابد أن تضع الأمر التالي في بداية النظام الذي يخدم شبكة اتصالات

SET EXCLUSIVE OFF

ومعناه أن كل الملفات التي يشتمل عليها النظام يستطيع أكثر من مستفيد داخل الشبكة فتحها والقيام بوظائف قواعد البيانات المطلوبة. ولابد أيضا تبعا لذلك أن يتم غلق الملف أو السجل عندما يرغب أحد المستفيدين في تعديل محتويات هذا الملف أو هذا السجل.

ملاحـظة: استخـدام أمـر USE <file> EXCLUSIVE يفتـح الملف المـذكـور في الأمر ليستخدم استخداما منفردا.

وقد شرحنا في الفصل الحادي عشر أن حزمة «كلبر» تشتمل ضمن ملفاتها على ملف باسم LOCKS.PRG. وهذا الملف يشتمل على ٤ وظائف خاصة تتحكم في غلق السجل أو الملف قبل التعامل معه في شبكة الاتصالات وهذه الوظائف هي:

NET_USE() - FIL_LOCK() - REC_LOCK() - ADD_REC()

وشرحنا أيضا هذه الوظائف الأربعة.

وتعتمـد فكـرة هذه الـوظـائف الأربعة على أن المستفيد يدخل معطيات قبل استدعاء إحداها تحدد عدد الثواني التي سيستمر خلالها النظام في محاولة فتح السجل

أو الملف قبل أن ينهي البرنامج . إذ ربما أن الملف أو السجل أغلق بواسطة مستفيد آخر في محطة أخرى لاجراء عملية صغيرة لمدة قليلة وسيعود بعدها إلى وضعه الطبيعي .

وبإمكانك استخدام هذه الوظائف كما هي داخل النظام ويلزمك فقط إضافة برنامج LOCKS.PRG للنظام ليتم ترجمته وربطه مع النظام. كما يمكنك تعديل هذه الوظائف لتعطي مرونة لمستخدم النظام تتمثل في إظهار رسالة له في حالة فشل فتح الملف أو السجل. وتعطيه الفرصة لتكرار محاولة غلق الملف أو السجل لمدة معينة أو الخروج من البرنامج .

ملاحظة : راجع هذه الوظائف في الفصل الحادي عشر جيدا لتفهم المنطق الذي أعدت به .

فمثلا الوظيفة ()NET_USE الموجودة في ملف LOCKS.PRG تحاول فتح الملف (جماعيا أو فرديا) لمدة يحددها المستفيد عند استدعاء الوظيفة إلا أنها في حالة الفشل لا تعطي المستفيد معلومات وافية توضح له ما حصل بالضبط.

وقد اخترنا إضافة تعديلات طفيفة على الوظائف الأربعة لكي يظهر للمستفيد رسالة واضحة في حالة فشل فتح السجل أو الملف بدلا من إغلاق الملفات وإنهاء النظام بالطريقة التقليدية كما يلي:

```
CLEAR
CLEAR ALL
RETURN
```

ويشتمل برنامج SAUDFS.PRG على الوظائف الخاصة التي تقابل الوظائف الخاصة. الموجودة في برنامج LOCKS.PRG الذي شرحناه في الفصل الحادي عشر. وهذه الوظائف الخاصة هي التي سنستخدمها داخل النظام الذي بين أيدينا ونوضح فيما يلي الاختلاف بين منطق كل من البرنامجين. برنامج LOCKS.PRG وبرنامج SAUDFS.PRG في البرنامج الأول يحدد المستفيد عدد الثواني التي سيستمر النظام خلالها في محاولة إعادة فتح الملف أو السجل فإذا لم يفتح في خلال هذه المدة ينتهي البرنامج .

أما في برنامجنا فإن النظام سيستمر لمدة خمس ثوان فقط في هذه المحاولة. فإذا فشلت المحاولة خلالها تظهر للمستفيد رسالة تسأله: هل ترغب في إعادة محاولة فتح السجل أو الملف لمدة خمس ثوان أخرى وبناء على إجابته تتكرر المحاولة لمدة جديدة أو ينتهي البرنامج.

ولذلك فإننا لا نحدد في المعطيات التي تدخل للوظيفة عدد الثواني التي ستتكرر فيها محاولة فتح الملف أو السجل. (راجع برنامج SAUDFS.PRG في شكل ٣٧-١٢).

الفرق بين النظام الفردي ونظام شبكة الاتصالات

تشتمل الأشكال من ٣٢-١٢ إلى ٣٧-١٢ على برامج نظام فواتير البيع الذي شرحناه قبل قليل مع إضافة بعض التعديلات التي تتطلبها ضرورة تشغيل النظام مع شبكة الاتصالات. وهذه الاضافات أو التعديلات هي:

١ - أضفنا حرفي N- لأسماء البرامج السابقة لنوضح أن هذه البرامج هي نفسها التي شرحناها ولكنها تستخدم مع شبكة اتصالات (Network).

٢ - أضفنا برنامجا جديدا للوظائف الجديدة في آخر شكل هو SAUDFS.PRG

٣ - أضفنا ملفا جديدا لملفات قاعدة البيانات. باسم FLAG.DBF وهذا الملف يستخدم مثل إشارة المرور التي تسمح بالسير إذا كانت خضراء أو بالتوقف إذا كانت حمراء وتوضيح ذلك أننا نضع رقم آخر فاتورة أصدرت على ملف ذاكرة (Memory file) وعندما نريد إصدار فاتورة جديدة فإننا نضع هذا الملف بالذاكرة لنجمع واحدا على آخر رقم ونخصص الرقم الجديد لرقم الفاتورة ثم نعيد الرقم الجديد إلى ملف MEM. الذي يوضع على القرص. والمشكلة هنا إذا استدعي أكثر من مستفيد في أكثر من محطة ملف SAINVNUM.MEM في نفس اللحظة. إذ يحتمل أن تأخذ الفاتورتان نفس الرقم. ولما كانت «كلبر» لا تعطينا إمكانية غلق ملف MEM. قبل استدعائه مثل ملف DBF. فإننا نستخدم ملف FLAG.DBF ليكون هو الاشارة التي تضيىء أخضر. فإذا فتح ملف FLAG.DBF بنجاح فسيتم تحميل ملف MEM. أما إذا فشل فتح ملف FLAG.DBF فستكون الاشارة

حمراء وبالتالي لن يتم تحميل ملف MEM. وبذلك نضمن أن مستفيدا واحدا داخل الشبكة يضيف رقما لآخر رقم فاتورة في نفس اللحظة.

ولذلك فإن ملف FLAG.DBF ملف يشتمل على حقل واحد فقط لأننا لا نتعامل مع بياناته. ويمكنك اختيار اسم وطول ونوع هذا الحقل.

٤ ـ لاحظ تكرار مثل هذه الأوامر داخل النظام عندما نريد فتح أحد الملفات

```
IF USE_NET ("cust",.F.)
        SET INDEX TO CUST
ELSE
        CLOSE DATABASES
        RETURN
ENDIF
```

وهذه الأوامر تقابل الأمرين التاليين في النظام الفردي السابق

```
USE cust
SET INDEX To cust
```

والسبب في هذا الاختلاف أننا في شبكة الاتصالات لا نريد فتح ملف الفهرس إذا فشل فتح ملف قاعدة البيانات. لأن «كلبر» إذا لم تتمكن من فتح ملف قاعدة البيانات (.DBF) وأرادت فتح ملف الفهرس (.NTX) فستحصل على رسالة خطأ. ونحن بهذه الطريقة نضمن عدم ظهور هذه الرسالة. وقلنا قبل قليل أن الوظيفة ()USE_NET تتولى فتح الملف ليستخدم بصفة فردية وتعطى فرصة تكرار المحاولة أما في النظام الذي يخدم مستفيدا واحدا فليس هناك احتمال أن تفشل «كلبر» في فتح الملف.

٥ ـ أضفنا وظيفة خاصة جديدة في نهاية برنامج SAINV.PRG باسم ()Flag_on وهذه الوظيفة تستخدم ملف FLAG.DBF لحماية رقم الفاتورة من التكرار. ويتم استدعاء هذه الوظيفة دائما قبل إصدار أي فاتورة جديدة.

```
 1 * ---------------------------------------------------------------------*
 2 * Program    : SAMAIN N.PRG                                             *
 3 * Author     : Magdi M. Abu Al-Ata                                      *
 4 * Date       : October 24, 1990                                         *
 5 * Purpose    : Main Module for main menu                                *
 6 *              Used for networking application                          *
 7 * Parameters : From DOS, you can optionally pass any parameter          *
 8 *              to run in color mode.                                    *
 9 *              missing a parameter will run in  monochrome mode         *
10 * ---------------------------------------------------------------------*
11 PUBLIC hd1,hd2
12 IF PCOUNT() > 0
13 |     PUBLIC syscolor
14 |     syscolor = "W+/B+,GR+/R+"
15 |     SETCOLOR(syscolor)
16 ENDIF
17 *  Set clipper environment
18 CLEAR
19 SET SCOREBOARD OFF
20 SET EXCLUSIVE OFF          && More than one person on the network
21                            && can open the file and do his work
22 SET WRAP ON
23 SET ESCAPE OFF
24 SET MESSAGE TO 24 CENTER
25 SET DELETED ON             && To ignore records marked for deletion
26 SET CONFIRM ON
27 SET DATE BRITISH
28 SET CENTURY ON
29 * Save the DOS screen
30 SAVE SCREEN TO dos_scr
31 CLEAR
32 dubl = CHR(201)+CHR(205)+CHR(187)+CHR(186)+CHR(188)+;
32  CHR(205)+CHR(200)+ CHR(186)+CHR(176)
33 @  2, 14 TO 5, 66 DOUBLE
34 hd1 =  "Company name: Arabic Computer"
35 hd2 =  "Copywrite: Magdi M. Abu Al-Ata 1990"
36 @ 3,t_cent(hd1,80) SAY hd1
37 @ 4,t_cent(hd2,80) SAY hd2
38 @ 0,0
39 @ 0,2  SAY "Date: "+ DTOC(DATE())
40 @ 0,65 SAY "Time: "+ TIME()
41
42 @  8, 14, 22, 66 BOX dubl
43 @  9,t_cent("MAIN MENU",80) SAY "MAIN MENU"
44 @ 10, 15 TO 10, 65
45 DO WHILE .T.
46 |      @ 12,30 PROMPT "Customer maintenance"  ;
46 |         MESSAGE "Maintain customer data "
47 |      @ 14,30 PROMPT "  Process invoices  "  ;
47 |         MESSAGE "Add, Edit, and Delete invoices"
48 |      @ 16,30 PROMPT "       Reports       "  ;
48 |         MESSAGE "Send reports to the printer"
49 |      @ 18,30 PROMPT " Files maintenance  "  ;
```

شكل ٣٢ - ١٢ برنامج SAMAIN_N. PRG

```
49 |      MESSAGE "Index, Back up, Restore and Pack files"
50 |    @ 20,30 PROMPT "          Exit          " ;
50 |      MESSAGE "Quit to DOS"
51 |    MENU TO choice1
52 |    DO CASE
53 |    CASE choice1 = 1
54 |    |    SAVE SCREEN TO menu_scr
55 |    |    menu_clr = SETCOLOR ()
56 |    |    @24,0 CLEAR
57 |    |    DO sacust_n
58 |    |    SETCOLOR (menu_clr)
59 |    |    RESTORE SCREEN FROM menu_scr
60 |    CASE choice1 = 2
61 |    |    SAVE SCREEN TO menu_scr
62 |    |    menu_clr = SETCOLOR()
63 |    |    @24,0 CLEAR
64 |    |    DO sainv_n
65 |    |    SETCOLOR (menu_clr)
66 |    |    RESTORE SCREEN FROM menu_scr
67 |    CASE choice1 = 3
68 |    |    SAVE SCREEN TO menu_scr
69 |    |    menu_clr = SETCOLOR ()
70 |    |    @24,0 CLEAR
71 |    |    DO sarep_n
72 |    |    SETCOLOR (menu_clr)
73 |    |    RESTORE SCREEN FROM menu_scr
74 |    CASE choice1 = 4
75 |    |    SAVE SCREEN TO menu_scr
76 |    |    menu_clr = SETCOLOR()
77 |    |    @24,0 CLEAR
78 |    |    DO sautil_n
79 |    |    SETCOLOR (menu_clr)
80 |    |    RESTORE SCREEN FROM menu_scr
81 |    CASE choice1 = 5
82 |<-------EXIT
83 |    ENDCASE
84 ENDDO
85 RESTORE SCREEN FROM dos_scr
86 SET COLOR TO
87 QUIT
88
89 *************************
90 *  End of main program  *
91 *************************
92
93 *--------------------------------*
94 *   Procedures and Functions     *
95 *   Called only by SAMAIN_N.PRG  *
96 *--------------------------------*
97
98 *****************
99 *  Function       : t_cent()
100 *  Purpose        : Displays text centered on the screen
```

تابع شكل ٣٢ - ١٢

```
101 *  Syntax           : t_cent(tstring, tlen)
102 *  Parameters       :
103 *      tstring      : Text to be displayed
104 *      tlen         : The width you need text to be centered on
105 *  Example          : t_cent( "Clipper world!", 80 )
106 ****************
107 FUNCTION t_cent
108 PRIVATE tstring, tlen
109 PARAMETERS tstring, tlen
110 STORE INT(TLEN/2 - (LEN(TSTRING)/2)) TO bigcol
111 RETURN bigcol
112
113 ****************************
114 *  End of file SAMAIN_N.PRG *
115 ****************************
```

تابع شكل ٣٢-١٢

```
 1 * --------------------------------------------------------------*
 2 * Program     : SACUST_N.PRG                                    *
 3 * Author      : Magdi M. Abu Al-Ata                             *
 4 * Date        : October 24, 1990                                *
 5 * Purpose     : To maintain customer files                      *
 6 *             : Used for networking application                 *
 7 * Called from: SAMAIN_N.PRG                                     *
 8 * --------------------------------------------------------------*
 9 @ 1,0 CLEAR TO 24,79
10 DO WHILE .T.
11 |     @ 1,14 TO 3,66 DOUBLE
12 |     @ 6,10 TO 22,70 DOUBLE
13 |     @ 7,25 SAY "Customer main data"
14 |     @ 8,11 TO 8,69
15 |     DO cust_fmt
16 |     @ 2,20 PROMPT "Add"  MESSAGE "Add a new customer to the file"
17 |     @ 2,29 PROMPT "Edit" MESSAGE "Modify customer record"
18 |     @ 2,41 PROMPT "Delete"  MESSAGE "Delete a customer record"
19 |     @ 2,52 PROMPT "Return"  MESSAGE "Return to the main menu"
20 |     MENU TO choice3
21 |     DO CASE
22 |     CASE choice3 = 0
23 <--------RETURN
24 |    CASE choice3 = 1
25 |    |     DO sacustad
26 |    CASE choice3 = 2
27 |    |     DO sacusted
28 |    CASE choice3 = 3
29 |    |     DO sacustde
30 |    CASE choice3 = 4
31 <--------RETURN
```

شكل ٣٣-١٢ SACUST_N.PRG

```
32 |     ENDCASE
33 ENDDO
34
35 ************************
36 *  End of program      *
37 ************************
38
39 *--------------------------------*
40 *   Procedures and Functions     *
41 *   Called only by SACUST_N.PRG  *
42 *--------------------------------*
43
44 ****************
45 *  Procedure    : sacustad
46 *  Purpose      : Adds new customer
47 ****************
48 PROCEDURE sacustad
49 @24,0 CLEAR
50 IF use_net ("custmr",.F.)        && Open CUSTMR.DBF
51 |    SET INDEX TO custmr
52 ELSE
53 <----RETURN
54 ENDIF
55 DO WHILE .T.
56 |    SET ESCAPE ON
57 |    @ 24,24 SAY "Press Esc to return to the menu"
58 |    @ 10,27 GET m_acct PICTURE '9999' VALID new_rec()
59 |    READ
60 |    @ 24,0
61 |    IF LASTKEY() = 27
62 |    |    CLOSE DATABASES
63 |    |    @ 24, 0
64 <---------RETURN
65 |    ENDIF
66 |    SET ESCAPE OFF
67 |    @ 12, 27 GET m_fnam
68 |    @ 12, 41 GET m_mnam
69 |    @ 12, 54 GET m_lnam
70 |    @ 14, 27 GET m_company
71 |    @ 16, 27 GET m_addr
72 |    @ 18, 27 GET m_city
73 |    @ 20, 27 GET m_phone PICTURE "@K 999-9999"
74 |    READ
75 |    REPLACE company with m_company, fnam with m_fnam, ;
75 |      mnam with m_mnam, lnam with m_lnam, addr with m_addr, ;
75 |      city with m_city , phone with m_phone
76 |    STORE " " TO m_ok
77 |    @ 24, 20 SAY "Enter another new customer? (Y/N)---) "GET m_ok
78 |    READ
79 |    IF UPPER (m_ok) = "Y"
80 |    |    @ 24,0 CLEAR
81 |    |    UNLOCK
82 |    |    DO cust_fmt
```

تابع شكل ٣٣ - ١٢

```
 83 |     ELSE
 84 |     |    CLEAR
 85 |     |    CLOSE DATABASES
 86 <---------RETURN
 87 |     ENDIF
 88 ENDDO
 89
 90 ****************
 91 *  Procedure    : sacusted
 92 *  Purpose      : Edits an existing customer
 93 ****************
 94 PROCEDURE sacusted
 95 @24,0 CLEAR
 96 IF use_net ("custmr",,F.)
 97 |     SET INDEX TO custmr
 98 ELSE
 99 <----RETURN
100 ENDIF
101 DO WHILE .T.
102 |     SET ESCAPE ON
103 |     @ 24,24 SAY "Press Esc to return to the menu"
104 |     @ 10,27 GET m_acct PICTURE "@K 9999" VALID rec_fnd()
105 |     READ
106 |     @ 24,0
107 |     IF LASTKEY() = 27
108 |     |    CLOSE DATABASES
109 |     |    @ 24, 0
110 <---------RETURN
111 |     ENDIF
112 |     SET ESCAPE OFF
113 |     IF m_acct <> SPACE(4)
114 |     |    @ 12,27 GET fnam
115 |     |    @ 12,41 GET mnam
116 |     |    @ 12,54 GET lnam
117 |     |    @ 14,27 GET company
118 |     |    @ 16,27 GET addr
119 |     |    @ 18,27 GET city
120 |     |    @ 20,27 GET phone PICTURE "@K 999-9999"
121 |     |    READ
122 |     ELSE
123 |<-------EXIT
124 |     ENDIF
125 |     STORE " " TO m_ok
126 |     @ 24, 0
127 |     @ 24, 20 SAY "Edit another customer record? (Y/N)----> " GET m_ok
128 |     READ
129 |     IF UPPER (m_ok) = "Y"
130 |     |    @24,0
131 |     |    UNLOCK
132 |     |    DO cust_fmt
133 |     ELSE
134 |<-------EXIT
135 |     ENDIF
136 ENDDO
```

تابع شكل ٣٣-١٢

```
137 CLEAR
138 CLOSE DATABASES
139 RETURN
140
141 *****************
142 *  Procedure    : sacustde
143 *  Purpose      : Deletes an existing customer
144 *****************
145 PROCEDURE sacustde
146 @24,0
147 IF use_net ("custmr",.F.)
148 |     SET INDEX TO custmr
149 ELSE
150 <----RETURN
151 ENDIF
152 DO WHILE .T.
153 |     SET ESCAPE ON
154 |     @ 24,24 SAY "Press Esc to return to the menu"
155 |     @ 10,27 GET m_acct PICTURE "@K 9999" VALID rec_fnd()
156 |     READ
157 |     @ 24,0
158 |     IF LASTKEY() = 27
159 |     |    CLOSE DATABASES
160 <--------RETURN
161 |     ENDIF
162 |     SET ESCAPE OFF
163 |     @ 12,27 GET fnam
164 |     @ 12,41 GET mnam
165 |     @ 12,54 GET lnam
166 |     @ 14,27 GET company
167 |     @ 16,27 GET addr
168 |     @ 18,27 GET city
169 |     @ 20,27 GET phone PICTURE "@K 999-9999"
170 |     CLEAR GETS
171 |     IF .NOT. EOF()
172 |     |    STORE " " TO m_ok
173 |     |    @24, 20 SAY "Delete this customer record? (Y/N)---->" GET m_ok
174 |     |    READ
175 |     |    IF UPPER (m_ok) = "Y"
176 |     |    |    DELETE
177 |     |    |    @24,0
178 |     |    |    UNLOCK
179 |     |    |    DO cust_fmt
180 |     |    |    STORE " " TO m_ok
181 |     |    |    @ 24, 20 SAY "Delete another customer record? (Y/N)----> "GET m_ok
182 |     |    |    READ
183 |     |    |    IF UPPER (m_ok) = "Y"
184 |     |    |    |    @24,0
185 |     |    |    |    UNLOCK
186 |     |    |    |    DO cust_fmt
187 ^----------------LOOP
188 |     |    |    ENDIF
189 |     |    ENDIF
190 |     ENDIF
```

تابع شكل ٣٣ - ١٢

```
191 |     CLEAR
192 |     CLOSE DATABASES
193 <----RETURN
194 ENDDO
195
196 ****************
197 *  Procedure    : cust_fmt
198 *  Purpose      : Drows customer screen format
199 ****************
200 PROCEDURE cust_fmt
201 * Make the variables globally available to the subroutines.
202 PUBLIC m_acct, m_company, m_addr, m_fnam, m_mnam, m_lnam, ;
202  m_city,  m_phone
203 STORE SPACE (4) to m_acct
204 STORE SPACE (20) to m_company, m_addr
205 STORE SPACE (12) to m_fnam, m_mnam, m_lnam
206 STORE SPACE (17) to m_city
207 STORE SPACE (8) to m_phone
208 @ 10, 12 SAY "Account No.  :"
209 @ 10, 27 SAY m_acct
210 @ 12, 12 SAY "Customer name:"
211 @ 12; 27 GET m_fnam
212 @ 12, 41 GET m_mnam
213 @ 12, 54 GET m_lnam
214 @ 14, 12 SAY "Company Name :" GET m_company
215 @ 16, 12 SAY "Address      :" GET m_addr
216 @ 18, 12 SAY "City         :" GET m_city
217 @ 20, 12 SAY "Phone        :" GET m_phone PICTURE "999-9999"
218 CLEAR GETS
219
220 ****************
221 *  Function     : new_rec
222 *  Purpose      : Confirms that the account no. not exist
223 *  Syntax       : new_rec()
224 *  Returns      : .T. or .F.
225 *  Example      : IF new_rec()
226 ****************
227 FUNCTION new_rec
228 IF m_acct = SPACE(4)
229 |     TONE (349,3)
230 |     TONE (300,5)
231 |     @ 24,0
232 |     @ 24,25 SAY "Press Esc to go back to menu."
233 <----RETURN .F.
234 ENDIF
235 SEEK &m_acct
236 IF .NOT. EOF()
237 |     TONE (349,3)
238 |     @ 24,0
239 |     @ 24,26 SAY "Duplicate Number. Try Again."
240 <----RETURN .F.
241 ELSE
242 |     IF rec_add()
```

تابع شكل ٣٣-١٢

```
243 |   |     @ 24,0
244 |   |     REPLACE acct with m_acct
245 <--------RETURN .T.
246 |     ELSE
247 <--------RETURN .F.
248 |     ENDIF
249 ENDIF
250
251 ****************
252 *  Function      : rec_fnd
253 *  Purpose       : Validation for edit and delete processing
254 *  Returns       : .T. or .F.
255 *  Syntax        : rec_fnd ()
256 *  Purpose       : IF rec_fnd
257 ****************
258 FUNCTION rec_fnd
259 IF m_acct = SPACE(4)
260 |     GO BOTTOM
261 |     SKIP
262 <----RETURN .T.
263 ENDIF
264 SEEK &m_acct
265 IF EOF()
266 |     TONE(349,3)
267 |     @ 24,22 SAY "Invalid account number. Try again."
268 <----RETURN .F.
269 ELSE
270 |     IF lock_rec()
271 |   |     @ 24,0
272 <--------RETURN .T.
273 |     ELSE
274 <--------RETURN .F.
275 |     ENDIF
276 ENDIF
277
278 *****************************
279 *  End of file SACUST_N.PRG *
280 *****************************
```

تابع شكل ٣٣-١٢

```
 1 * -------------------------------------------------------------------*
 2 * Program     : SAINV_N.PRG                                           *
 3 * Author      : Magdi M. Abu Al-Ata                                   *
 4 * Date        : October 24, 1990                                      *
 5 * Purpose     : To process sales invoices                             *
 6 *             : Used for networking application                       *
 7 * Called from: SAMAIN_N.PRG                                           *
 8 * -------------------------------------------------------------------*
 9 @1,0 CLEAR TO 24,79
10 DO WHILE .T.
11 |    @  1,14 TO 3, 66  DOUBLE
12 |    @  9,1 TO 23,78  DOUBLE
13 |    @ 11,2 TO 11,77
14 |    DO inv_fmt
15 |    @ 2,20 PROMPT "Add"  MESSAGE "Create a new invoice"
16 |    @ 2,29 PROMPT "Modify"  MESSAGE "Edit an existing invoice"
17 |    @ 2,41 PROMPT "Delete"  MESSAGE "Delete an invoice"
18 |    @ 2,52 PROMPT "Return" MESSAGE "Go to main menu"
19 |    MENU TO choice2
20 |    DO CASE
21 |    CASE choice2 = 0
22 <--------RETURN
23 |    CASE choice2 = 1
24 |    |    DO sainvadd
25 |    CASE choice2 = 2
26 |    |    DO sainvedt
27 |    CASE choice2 = 3
28 |    |    DO sainvdel
29 |    CASE choice2 = 4
30 <--------RETURN
31 |    ENDCASE
32 ENDDO
33
34 ************************
35 *  End of program      *
36 ************************
37
38 *--------------------------------*
39 *   Procedures and Functions     *
40 *   Called only by SAINV_N.PRG   *
41 *--------------------------------*
42
43 *****************
44 *  Procedure     : sainvadd
45 *  Purpose       : Creates a new invoice
46 *****************

47 PROCEDURE sainvadd
48 @24,0
49 PUBLIC m_dup
50 m_dup = .F.
51 IF use_net ("custmr",,.F.)
52 |    SET INDEX TO custmr
53 ELSE
```

تابع شكل ٣٤-١٢ SAINV_N.PRG

```
54 |    CLOSE DATABASES
55 <----RETURN
56 ENDIF
57 SELECT 0
58 IF use_net ("invoice",.F.)
59 |  . SET INDEX TO invoice
60 ELSE
61 |    CLOSE DATABASES
62 <----RETURN
63 ENDIF
64 SELECT 0
65 IF use_net("inv_item",.F.)
66 |    SET INDEX TO inv_item
67 ELSE
68 |    CLOSE DATABASES
69 <----RETURN
70 ENDIF
71 SELECT 0
72 IF use_net("flag",.F.)
73 |    * GOTO 1
74 |    @ 24,0
75 ELSE
76 |    CLOSE DATABASES
77 <----RETURN
78 ENDIF
79 DO WHILE .T.
80 |     IF flag_on()
81 |     |   @ 24,0
82 |     ELSE
83 |     |   CLOSE DATABASES
84 <---------RETURN
85 |     ENDIF
86 |     SET ESCAPE ON
87 |     @ 5, 18 GET m_acct PICTURE "9999" VALID val_get()
88 |     READ
89 |     IF LASTKEY() = 27
90 |     |   CLOSE DATABASES
91 |     |   @24,0
92 <---------RETURN
93 |     ENDIF
94 |     SET ESCAPE OFF
95 |     @ 5, 62 SAY m_inv PICTURE '999999'
96 |     @ 6, 18 SAY m_sal_dat
97 |     @ 7, 18 SAY TRIM(custmr->fnam) + " "  + TRIM(custmr->mnam) + ;
97 |       " " + custmr->lnam
98 |     @  8, 18 SAY TRIM(custmr-> company) + "," + ;
98 |       TRIM(custmr->addr) + "," + TRIM (custmr->city) + "."
99 |     @  6,62 GET m_sal_man
100 |    READ
101 |    REPLACE invoice-> acct_no with m_acct, invoice->inv with m_inv,;
    |    invoice->sal_dat with m_sal_dat, invoice->sal_man with m_sal_m an
101 |    REPLACE invoice-> acct_no with m_acct, invoice->inv with m_inv,;
         invoice->sal_dat with m_sal_dat, invoice->sal_man with m_sal_m an
```

تابع شكل ٣٤ـ١٢

```
102 |      DO inv_data
103 |      STORE " " TO m_ok
104 |      @ 24,0
105 |      @ 24,25 SAY "Print this invoice? [Y/N]---> " GET m_ok
106 |      READ
107 |      IF UPPER (m_ok) = "Y"
108 |     |    DO inv_prn
109 |      ENDIF
110 |      STOR " " TO m_ok
111 |      @ 24, 0
112 |      @ 24, 20 SAY "Enter another new invoice? (Y/N)---> "GET m_ok
113 |      READ
114 |      @ 12,3 clear to 22,76
115 |      @24,0
116 |      IF UPPER (m_ok) = "Y"
117 |     |    UNLOCK
118 |     |    DO inv_fmt
119 |      ELSE
120 |     |    CLOSE  DATABASES
121 <---------RETURN
122 |      ENDIF
123 ENDDO
124
125 ################
126 #  Procedure     : sainvedt
127 #  Purpose       : Modify an existing invoice
128 ################

129 PROCEDURE sainvedt
130 @ 24,0 CLEAR
131 PUBLIC m_dup
132 m_dup = .T.
133 IF use_net ("custmr", .F.)
134 |      SET INDEX TO custmr
135 ELSE
136 |      CLOSE DATABASES
137 <----RETURN
138 ENDIF
139 SELECT 0
140 IF use_net ("invoice",.F.)
141 |      SET INDEX TO invoice
142 ELSE
143 |      CLOSE DATABASES
144 <----RETURN
145 ENDIF
146 SELECT 0
147 IF use_net ("inv_item",.F.)
148 |      SET INDEX TO inv_item
149 ELSE
150 |      CLOSE DATABASES
151 <----RETURN
152 ENDIF
153 DO WHILE .T.
```

تابع شكل ٣٤-١٢

```
154 |     SET ESCAPE ON
155 |     @ 5, 62 GET m_inv PICTURE '999999' VALID val_seek()
156 |     READ
157 |     IF LASTKEY() = 27
158 |     |   CLOSE DATABASES
159 <---------RETURN
160 |     ENDIF
161 |     SET ESCAPE OFF
162 |     SELECT invoice
163 |     IF lock_rec()
164 |     |   @ 5, 18 SAY custmr->acct
165 |     |   @ 6, 18 SAY invoice->sal_dat
166 |     |   @ 6, 62 SAY invoice->sal_man
167 |     |   @ 7, 18 SAY TRIM(custmr->fnam) + " "  + ;
167 |     |     TRIM(custmr->mnam) + " " + custmr->lnam
168 |     |   @  8, 18 SAY TRIM(custmr-> company) + "," + ;
168 |     |     TRIM(custmr->addr) + "," + TRIM (custmr->city) + "."
169 |     ELSE
170 |     |   CLOSE DATABASES
171 <---------RETURN
172 |     ENDIF
173 |     * This next section prepares for the DEBIT().
174 |     DECLARE ar_edit[5]
175 |     ar_edit[1] = "item_no"
176 |     ar_edit[2] = "qty"
177 |     ar_edit[3] = "desc"
178 |     ar_edit[4] = "unit_price"
179 |     ar_edit[5] = "qty * unit_price"
180 |     @15,6 CLEAR TO 19,73
181 |     @21,6 CLEAR TO 21,73
182 |     SELECT inv_item
183 |     SEEK m_inv
164 |     IF EOF()
185 |     |   TONE(349,3)
186 |     |   TONE(300,5)
187 |     |   @ 24,0
188 |     |   @ 24,20 SAY "ERROR. No invoice items exist."
189 |     |   TONE(300,5)
190 |     |   TONE(349,3)
191 |     |   INKEY(3)
192 |     |   CLOSE DATABASES
193 |     |   @ 24, 0
194 |     |   DO inv_fmt
195 <---------RETURN
196 |     ENDIF
197 |     temp = TRIM(STR(SECONDS(),5,0))  && 5 is all you can get
198 |     COPY TO &temp WHILE inv_no = m_inv
199 |     SELECT 0
200 |     IF use_net(temp,.T.) .AND. lock_fil()
201 |     |   dummy = ""
202 |     |   SET DELETED OFF
203 |     |   STORE 0 TO m_tot
204 |     |   @ 24,0
205 |     |   @ 24,27 SAY "Select field and press ENTER."
```

تابع شكل ٣٤-١٢

```
206 |  |    DBEDIT (12,3,19,76,ar_edit,"udf_edit",dummy,  dummy, dummy, SPACE(4))
207 |  |    @ 24,0
208 |  |    SET DELETED ON
209 |  |    USE                    && Close the temp. file
210 |  |    SELECT inv_item
211 |  |    IF lock_fil()
212 |  |   |    DELETE ALL FOR inv_no=m_inv
213 |  |    ELSE
214 |  |   |    @22,0 CLEAR
215 |  |   |    @ 24, 5 SAY "File " + TRIM(&temp) + ".DBF has some problem."
216 <-----------RETURN
217 |  |    ENDIF
218 |  |    APPEND FROM &temp
219 |  |    temp = TRIM(temp)+".DBF"
220 |  |    * ERASE &temp          && Invalid syntax
221 |  |    ERASE (temp)
222 |  |    DO inv_prn
223 |  |    CLOSE DATABASES
224 |  |    DO inv_fmt
225 <--------RETURN
226 |    ELSE
227 |  |    CLOSE DATABASES
228 <--------RETURN
229 |    ENDIF
230 ENDDO
231
232 *****************
233 *  Procedure    : sainvdel
234 *  Purpose      : Deletes an existing invoice
235 *****************
236 PROCEDURE sainvdel
237 @ 24,0
238 IF use_net ("custmr",.F.)
239 |    SET INDEX TO custmr
240 ELSE
241 |    CLOSE DATABASES
242 <----RETURN
243 ENDIF
244 SELECT 0
245 IF use_net ("invoice",.F.)
246 |    SET INDEX TO invoice
247 ELSE
248 |    CLOSE DATABASES
249 <----RETURN
250 ENDIF
251 SELECT 0
252 IF use_net("inv_item",.F.)
253 |    SET INDEX TO inv_item
254 ELSE
255 |    CLOSE DATABASES
256 <----RETURN
257 ENDIF
258 rno = RECNO()
259 SET ESCAPE ON
```

تابع شكل ٣٤-١٢

```
260 @ 5, 62 GET m_inv PICTURE '999999' VALID val_seek()
261 READ
262 IF LASTKEY() = 27
263 |     CLOSE DATABASES
264 <----RETURN
265 ENDIF
266 SET ESCAPE OFF
267 IF lock_rec()
268 |     @ 5, 18 SAY custmr->acct
269 |     @ 6, 18 SAY invoice->sal_dat
270 |     @ 6,62 SAY invoice->sal_man
271 |     @ 7, 18 SAY TRIM(custmr->fnam) + " "  + TRIM(custmr->mnam) ;
271 |       + " " + custmr->lnam
272 |     @  8, 18 SAY TRIM(custmr-> company) + "," + ;
272 |       TRIM(custmr->addr) + "," + TRIM (custmr->city) + "."
273 ELSE
274 |     CLOSE DATABASES
275 <----RETURN
276 ENDIF
277 SELECT inv_item
278 SEEK m_inv
279 rno = RECNO()
280 IF EOF()
281 |     TONE (349,3)
282 |     TONE(300,5)
283 |     @ 24, 0
284 |     @ 24,27 SAY "ERROR. No invoice items exist.    "
285 |     TONE(300,5)
286 |     TONE(349,3)
287 ENDIF
288 line = 12
289 FOR i = 1 TO 10
290 @ line,  4 SAY item_no
291 @ line, 15 SAY STR(qty,4,0)
292 @ LINE, 25 SAY desc
293 @ line, 51 SAY STR(unit_price,7,2)
294 SKIP
295 IF inv_no = m_inv
296 |     line = line + 1
297 ELSE
298 |     EXIT
299 ENDIF
300 NEXT
301 STORE ' ' TO m_ok
302 @ 24,0
303 @ 24, 22 SAY "Delete this invoice? (Y/N)--> "GET m_ok
304 READ
305 IF UPPER (m_ok) = "Y"
306 |     GOTO rno
307 |     DO WHILE .T.
308 |     |     IF lock_rec() .AND. .NOT. EOF()
309 |     |     |     DELETE
310 |     |     ENDIF
311 |     |     SKIP
```

تابع شكل ٣٤ - ١٢

```
312 |    |__  IF .NOT. EOF() .AND. inv_no = m_inv
313 |    ^--------LOOP
314 |    |    ELSE
315 |    |    |    @ 24,0 CLEAR
316 |  <----------EXIT
317 |    |__  ENDIF
318 |    ENDDO
319 |    SELECT invoice
320 |    IF lock_rec()
321 |    |__  DELETE
322 |    ENDIF
323 ELSE
324 |    @ 24, 0
325 |    @ 24, 20 SAY "Invoice not deleted. Press any key."
326 |    INKEY (0)
327 |    @ 24,0
328 ENDIF
329 CLOSE DATABASES
330 RETURN
331
332 ****************
333 *  Procedure    : inv_data
334 *  Purpose      : Writes invoice data
335 ****************
336 PROCEDURE inv_data
337 line = 12
338 SELECT inv_item
339 DO WHILE .T.
340 |    @ 24,0
341 |    @ 24,20 SAY "Enter item code or oress ENTER to exit"
342 |    IF rec_add ()
343 |    |    @ line,    4 GET  m_item_no PICTURE '999'
344 |    |    READ
345 |    |    IF m_item_no = SPACE (3)
346 |    |    |    DELETE
347 |<-----------EXIT
348 |    |    ENDIF
349 |    |    @ 24,0
350 |    |    @ line,  15 GET  m_qty PICTURE '@z 99'
351 |    |    @ line,  25 GET  m_desc
352 |    |    @ line,  51 GET  m_unit_price PICTURE '@Z 9999.99'
353 |    |    READ
354 |    |    REPLACE inv_no with m_inv, item_no with m_item_no  desc ;
354 |    |     with m_desc, qty with m_qty, unit_price with m_unit_price
355 |    |    m_ext = m_qty * m_unit_price
356 |    |    m_tot = m_tot + m_ext
357 |    |    @ line, 64 SAY m_ext PICTURE '999999.99'
358 |    |    @ 21, 64 SAY "========="
359 |    |    @ 22, 40 SAY "Total"
360 |    |    @ 22, 64 SAY m_tot PICTURE '999999.99'
361 |    ELSE
362 <--------RETURN
363 |    ENDIF
364 |    .UNLOCK
```

تابع شكل ٣٤ - ١٢

```
365 |     IF line <> 20
366 |     |    line = line + 1
367 |     ELSE
368 |     |    SCROLL (12,3,20,76,1)
369 |     ENDIF
370 |     STORE 0 TO m_qty, m_unit_price, m_ext
371 |     STORE SPACE (3) TO m_item_no
372 |     STORE SPACE (20) TO m_desc
373 ENDDO
374 RETURN
375
376 *****************
377 *  Procedure     : inv_fmt
378 *  Purpose       : Drows an invoice screen format
379 *****************
380 PROCEDURE inv_fmt
381 PUBLIC m_acct, m_acct_no, m_addr, m_company, m_fnam, m_mnam, ;
381  m_lnam, m_city, m_phone, m_inv, m_qty, m_unit_price, m_sal_man, ;
381  m_sal_dat, m_item_no, m_desc, m_ext,  m_tot
382 * Then it gives those vairables their starting values.
383 STORE CTOD("  /  /  ") TO m_sal_dat
384 STORE 0 TO m_qty, m_unit_price, m_inv, m_ext, m_tot
385 STORE SPACE (12) TO m_mnam
386 STORE SPACE (12) TO m_sal_man
387 STORE SPACE (3)  TO  m_item_no
388 STORE SPACE (4)  TO m_acct, m_acct_no
389 STORE SPACE (14) TO m_phone
390 STORE SPACE (12) TO m_lnam
391 STORE SPACE (17) TO m_city
392 STORE SPACE (12) TO m_fnam
393 STORE SPACE (20) TO m_desc
394 STORE SPACE (20) TO m_company, m_addr
395 @ 5, 6 SAY "Account No:"
396 @ 5,18 SAY m_acct
397 @ 5,51 SAY "Invoice No:"
398 @ 5,62 SAY m_inv PICTURE "@Z 999999"
399 @ 6, 6 SAY "Sale Date :"
400 @ 6,18 SAY m_sal_dat
401 @ 6,51 SAY "Salesman  :"
402 @ 6,62 SAY m_sal_man
403 @ 7, 6 SAY "Sold to   :"
404 @ 7,18 SAY m_fnam
405 @ 7,25 SAY m_mnam
406 @ 7,27 SAY m_lnam
407 @ 8,18 SAY m_company
408 @ 8,39 SAY m_addr
409 @ 8,60 SAY m_city
410 @10, 4 SAY "Item No    Quan.    Description          Price         Amount"
411 @ 12,2 CLEAR TO 22,77  && NEW COMMAND to erase the items pained
412 RETURN
413
414 *****************
415 *  Procedure     : inv_prn
416 *  Purpose       : Print an invoice on a printer
417 *****************
```

تابع شكل ٣٤-١٢

```
418 PROCEDURE inv_prn
419 SET CURSOR OFF
420 SAVE SCREEN TO prin_inv
421 STORE 0 TO m_tot
422 @ 8,15 CLEAR TO 20,60
423 @ 8,15 TO 20,60 DOUBLE
424 @12,20 SAY "Ready to print this invoice, Make"
425 @14,20 SAY "Sure the printer is on and ready."
426 @16,20 SAY "Press any key to begin."
427 INKEY (0)
428 @ 9,16 CLEAR TO 19,59
429 @14,20 SAY"    Printing Invoice: " + STR(m_inv,6,0)
430 pline = 1
431 page = 1
432 SET DEVICE TO PRINT
433 IF m_dup
434 |    @pline + 2, t_cent("Modified Invoice" , 80) SAY  "Modified Invoice"
435 |    pline = pline + 2
436 ENDIF
437 hd1 = "Your company name"
438 hd2 = "Your Address and C.R."
439 @pline + 1, t_cent(hd1, 80) SAY hd1
440 @pline + 2, t_cent(hd2, 80) SAY hd2
441 pline = pline + 3
442 @pline + 1, 6 SAY "Sold To: " + TRIM(custmr->fnam) + " " ;
442   + TRIM(custmr->mnam) + " " + custmr->lnam
443 @pline + 1, 51 SAY "Account #: " + custmr->acct
444 @pline + 2, 15 SAY custmr->company
445 @pline + 2, 51 SAY "Invoice No: " + STR(m_inv,6,0)
446 @pline + 3, 15 SAY custmr->addr + ","
447 @pline + 3, 51 SAY "Sale date: " + DTOC(invoice->sal_dat)
448 @pline + 4, 15 SAY TRIM(custmr->city) + "."
449 @pline + 4, 51 SAY "Salesman: " + invoice->sal_man
450 @pline + 6, 5 SAY REPLICATE ("-",70)
451 pline = pline + 7
452 @pline + 1, 7 SAY "Item             Item            " + "Unit        Amount
453 @pline + 2, 7 SAY "Code    Quan.   Description      " + "Prince           "
454 @pline + 3, 5 SAY REPLICATE ("-",70)
455 pline = pline + 5
456 DO item_prn
457 pline = pline + 2
458 @pline + 3, 49 SAY "    Total: " + STR(m_tot,12,2)
459 EJECT
460 SET DEVICE TO SCREE
461 RESTORE SCREEN FROM prin_inv
462 SET CURSOR ON
463 RETURN
464
465 ****************
466 *  Procedure     : item_prn
467 *  Purpose       : Prints invoice items
468 ****************
469 PROCEDURE item_prn
470 SELECT inv item
```

تابع شكل ٣٤-١٢

```
471 SEEK m_inv
472 IF EOF()
473 |    @pline + 3, 7 SAY "This invoice has no items to print."
474 <----RETURN
475 ENDIF
476 DO WHILE .T.
477 |    IF pline > 55
478 |    |    EJECT
479 |    |    @pline + 1, 22 SAY hd1
480 |    |    @pline + 2, 22 SAY hd2
481 |    |    @pline + 3, 22 SAY "Page No. " + STR(page,2,0)
482 |    |    pline = pline + 6
483 |    |    @pline + 2, 51 SAY "Invoice No: " + str(m_inv,6,0)
484 |    |    @pline + 3, 51 SAY "Sale Date: " + DTOC(invoice->sal_dat)
485 |    |    @pline + 4, 51 SAY "Salesman: " + invoice->sal_man
486 |    |    @pline + 6, 5  SAY REPLICATE ("-",70)
487 |    |    pline = pline + 7
488 |    |    @pline + 1, 7 SAY "Item            Item                     Amou: "
489 |    |    @pline + 2, 7 SAY "Code    Quan.  Description " + "
490 |    |    @pline + 3, 5  SAY REPLICATE ("-",70)
491 |    |    pline = pline + 4
492 |    ENDIF
493 |    * Here the item information is printed.
494 |    @pline + 1, 7 SAY item_no
495 |    @pline + 1,15 SAY STR (qty,4,0)
496 |    @pline + 1,25 SAY desc
497 |    @pline + 1,51 SAY STR (unit_price,7,2)
498 |    @pline + 1,64 SAY STR (qty * unit_price,9,2)
499 |    m_tot = m_tot + (qty * unit_price)
500 |    pline = pline + 1
501 |    SKIP
502 |    IF .NOT. EOF() .AND. inv_no = m_inv
503 ^--------LOOP
504 |    ELSE
505 <--------RETURN
506 |    ENDIF
507 ENDDO
508
509 *****************
510 *  Function     : val_seek
511 *  Purpose      : To validate the entered invoice number
512 *  Returns      : .T. or .F.
513 *  Syntax.      : val_seek()
514 *  Purpose      : IF val_seek()
515 *****************
516 FUNCTION val_seek
517 SELECT invoice
518 SET INDEX TO invoice
519 IF m_inv = 0
520 |    TONE(349,3)
521 |    TONE(300,5)
522 |    @ 24,0
523 |    @24, 20 SAY "Press Esc to go back to menu."
524 <----RETURN .F.
```

تابع شكل ٣٤-١٢

```
525 ENDIF
526 *SEEK STR(m_inv,6,0)
527 SEEK m_inv
528 IF EOF()
529 |     TONE(349,3)
530 |     TONE(300,5)
531 |     @ 24,0
532 |     @24,20 SAY "Sorry. No such invoice. Try again."
533 <----RETURN .F.
534 ELSE
535 |     store acct_no to srch_acct
536 |     SELECT custmr
537 |     seek &srch_acct
538 |     IF EOF()
539 |     |     TONE (349,3)
540 |     |     TONE (300,5)
541 |     |     @ 24,0
542 |     |     @ 24,20 SAY "Error. Customer no on file. "
543 |     |     IF TYPE ("rno") <> "U"
544 <-------------RETURN .T.
545 |     |     ENDIF
546 <---------RETURN .F.
547 |     ENDIF
548 <----RETURN .T.
549 ENDIF
550
551 ****************
552 *  Function      : udf_edit
553 *  Purpose       : Edits invoice item
554 *  Parameters    :
555 *       db_mod   : The current mode
556 *       a_fldd   : The field that is highlighted
557 *  Returns       : The current mode and highlighted field
558 *  Example       : DBEDIT(12,3,19,76,ar_edit,"udf_edit")
559 ****************
560 FUNCTION udf_edit
561 PARAMETERS db_mod, fld
562 get_fld = ar_edit[fld]
563 DO CASE
564 CASE db_mod = 0
565 |     @ 20,2 TO 20,77 DOUBLE
566 |     @ 21,45 TO 22,45 DOUBLE
567 |     @ 21, 7 SAY  "F3: Delete/Undelete item."
568 |     @ 21,56 SAY  "F4: Add an item. "
569 |     @ 22, 7 SAY "Esc: End edit."
570 |     @ 20,33 SAY IIF(DELETED(), "** Deleted **",REPLICATE(CHR(205),13))
571 <----RETURN 1
572 CASE db_mod = 1 .or. db_mod = 2
573 |     ? REPLICATE(CHR(7),2)
574 <----RETURN 1
575 CASE LASTKEY() = -2
576 |     IF DELETED()
577 |     |     RECALL
578 |     ELSE
```

تابع شكل ٣٤-١٢

```
579 |   |    DELETE
580 |    ENDIF
581 <----RETURN 2
582 CASE LASTKEY() = -3
583 |     APPEND BLANK
584 |     REPLACE inv_no WITH m_inv
585 |     KEYBOARD CHR(1)+CHR(30)
586 <----RETURN 1
587 CASE LASTKEY() = 13 .AND. fld<> 5
588 |     SET CURSOR ON
589 |     @ROW(),COL() GET &get_fld
590 |     READ
591 |     cur_rec = RECNO()
592 |     SUM  (qty * unit_price) to m_tot
593 |     @22,56 SAY "Total:"
594 |     @22,64 SAY m_tot PICTURE '9999999.99'
595 |     GO cur_rec
596 |     SET CURSOR OFF
597 <----RETURN 2
598 CASE LASTKEY() = 27
599 <----RETURN 0
600 OTHERWISE
601 <----RETURN 1
602 ENDCASE
603
604 ****************
605 *  Function     : val_get
606 *  Purpose      : Validates the invoice number
607 *  Syntax       : val_get()
608 *  Returns      : .T. or .F.
609 *  Example      : IF val_get()
610 ****************
611 FUNCTION val_get
612 SELECT custmr
613 IF m_acct = SPACE (4)
614 |     TONE (349,3)
615 |     TONE (300,5)
616 |     @ 24,0
617 |     @ 24,27 SAY "Press Esc to go back to menu."
618 <----RETURN .F.
619 ENDIF
620 SEEK &m_acct
621 IF EOF()
622 |     TONE(349,3)
623 |     TONE(300,5)
624 |     @ 24,0
625 |     @ 24,20 SAY "Sory. No such account. Try again."
626 <----RETURN .F.
627 ELSE
628 |     select invoice
629 |     IF rec_add ()
630 |   |    @ 24, 0
631 <--------RETURN .T.
632 |     ELSE
```

تابع شكل ٣٤ـ١٢

```
633 |   |    @ 4, 20 SAY "Unable to create invoice record."
634 <--------RETURN.F.
635 |     ENDIF
636 ENDIF
637
638 ****************
639 *  Function     : flag_on
640 *  Purpose      : To lock the flag file before reading and using
641 *               : an invoice number
642 *  Syntax       : flag_on()
643 *  Returns      : .T. or .F.
644 *  Example      : IF flag_on()
645 ****************
646 FUNCTION flag_on
647 select flag
648 *GOTO 1
649 IF .NOT. FILE ("sainvnum.mem")
650 |    SAVE ALL LIKE m_inv  TO SAINVNUM
651 ENDIF
652 IF lock_fil ()
653 |    RESTORE FROM sainvnum ADDITIVE
654 |    m_inv = m_inv + 1
655 |    SAVE ALL LIKE m_inv TO sainvnum
656 |    UNLOCK
657 |    STORE DATE() to m_sal_dat
658 <----RETURN .T.
659 ELSE
660 |    @ 24,26 SAY "Invoice not available now."
661 <----RETURN .F.
662 ENDIF
663
664 ***************************
665 *  End of file SAINV_N.PRG *
666 ***************************
```

تابع شكل ٣٤-١٢

```
 1 * -------------------------------------------------------------*
 2 * Program     : SAREP.PRG                                      *
 3 * Author      : Magdi M. Abu Al-Ata                            *
 4 * Date        : October 24, 1990                               *
 5 * Purpose     : To list customers by name or account no.       *
 6 * Called from: SAMAIN.PRG                                      *
 7 * -------------------------------------------------------------*
 8 DO WHILE .T.
 9 |    @15, 50 CLEAR TO 23,79
10 |    @15, 50, 23, 79 BOX dubl
11 |    @16,57 SAY "List customers"
12 |    @17, 51 TO 17,78
```

شكل ٣٥-١٢ برنامج SAREP_.N PRG

```
13 |      * Create a sub menu.
14 |      @18,57 PROMPT "  By Name  "  ;
14 |       MESSAGE "List customers alphabetically on last name"
15 |      @20,57 PROMPT "By Accout #"  ;
15 |       MESSAGE "List customers by account number"
16 |      @22,57 PROMPT "   Return  "  ;
16 |       MESSAGE "Return to the previous menu"
17 |      MENU TO choice5
18 |      DO CASE
19 |      CASE choice5 = 0
20 <--------RETURN
21 |      CASE choice5 = 1
22 |      |    IF use_net("custmr", .F.)
23 |      |    |    SET INDEX TO name
24 |      |    |    DO rep_cust WITH "Customers quick report by name"
25 |      |    ELSE
26 |      |    |    CLOSE DATABASES
27 <------------RETURN
28 |      |    ENDIF
29 |      CASE choice5 = 2
30 |      |    IF use_net("custmr", .F.)
31 |      |    |    SET INDEX TO custmr
32 |      |    |    DO rep_cust WITH "Customers quick report by account no."
33 |      |    ELSE
34 |      |    |    CLOSE DATABASES
35 <------------RETURN
36 |      |    ENDIF
37 |      CASE choice5 = 3
38 <--------RETURN
39 |      ENDCASE
40 ENDDO
41
42 **************************
43 *  End of program        *
44 **************************
45
46 *--------------------------------*
47 *   Procedures and Functions     *
48 *   Called only by SAREP.PRG     *
49 *--------------------------------*
50
51 *****************
52 *  Procedure     : rep_cust
53 *  Purpose       : prints customers on terminal or printer
54 *****************
55 PROCEDURE rep_cust
56 PARAMETERS pghd
57 PRIVATE cho,pr,pgno,pghd,pghdl,goon
58 STORE 1 TO pgno
59 STORE REPLICATE(CHR(178), LEN(pghd)) TO pghdl
60 STORE '?' TO goon
61 STORE " " TO cho
62 SAVE SCREEN TO rep_scr
63 CLEAR
```

تابع شكل ٣٥-١٢

```
64 @ 02,13 SAY "Direct output to the Printer/Terminal? (P/T)"  GET cho
65 READ
66 DO CASE
67 CASE UPPER(cho) = "P"
68 |     DO rep_prn
69 CASE UPPER(cho) = "T"
70 |     DO rep_scr
71 ENDCASE
72 RESTORE SCREEN FROM rep_scr
73
74 *****************
75 *  Procedure      : rep_prn
76 *  Purpose        : prints customers on printer
77 *****************
78 PROCEDURE rep_prn
79 WAIT " The report will direct to the printer. Ready? [Y/N] " TO pr
80 IF UPPER(PR)="Y"
81 |     ? " Please wait ... Report printing "
82 |     SET DEVICE TO PRINT
83 |     SET CONSOLE OFF
84 |     DO WHILE .NOT. EOF()
85 |     |   @ 2,5 SAY "Page No: "+STR(pgno,3)
86 |     |   @ 3,5 SAY "Date: " + ctod(date())
87 |     |   @ 5,t_cent(pghd)  SAY  pghd
88 |     |   @ 6,t_cent(pghd1) SAY  pghd1
89 |     |   @ 8,1 SAY "Account #              Customer name  "
90 |     |   list next 50 acct+space(7)+trim(fnam)+trim(mnam)+lnam
91 |     |   IF EOF()
92 |     |   |   @ prow()+3, 30 SAY " *** End of data *** "
93 |     |   ENDIF
94 |     |   pgno = pgno+1
95 |     |   EJECT
96 |     ENDDO
97 |     SET DEVICE TO SCREEN
98 |     SET CONSOLE ON
99 |     ? " Report finished ... Remove paper "
100 ELSE
101 <----RETURN
102 ENDIF
103
104 *****************
105 *  Procedure      : rep_scr
106 *  Purpose        : prints customers on screen
107 *****************
108 PROCEDURE rep_scr
109 @ 1,0  CLEA TO 21,78
110 @ 2,1 SAY "Page No. "
111 @ 2,60 SAY "Date: " +dtoc (date())
112 @ 3,t_cent(pghd,80) SAY pghd
113 @ 4,t_cent(pghd1,80) SAY pghd1
114 @ 6,  0 SAY REPLI("-",78)
115 ?"Account #              Customer name  "
116 @ 8,  1 SAY REPLI("-",78)
```

تابع شكل ٣٥-١٢

```
117 DO WHILE .NOT. EOF()
118 |     @ 2,11 SAY STR(pgno,3)
119 |     @ 9,1 CLEAR to 21,79
120 |     @ 09, 1
121 |     LIST OFF NEXT 10 ACCT+SPACE(7)+TRIM(fnms)+" "+TRIM(mnam)+"."+ lnam
122 |     IF EOF()
123 |     |   @ 23,1 SAY " *** End of data *** Press any key "
124 |     |   INKEY(3)
125 <--------RETURN
126 |     ENDIF
127 |     pgno = pgno+1
128 |     @ 23,1 SAY "Continue ...?(Y/N)" GET goon
129 |     READ
130 |     IF UPPER(goon) = "N"
131 |<-------EXIT
132 |     ENDIF
133 ENDDO
134
135 ***************************
136 *  End of file SAREP.PRG    *
137 ***************************
```

تابع شكل ٣٥-١٢

```
 1 * ------------------------------------------------------------------------*
 2 * Program     : SAUTIL_N.PRG                                               *
 3 * Author      : Magdi M. Abu Al-Ata                                        *
 4 * Date        : Nov., 12, 1990                                             *
 5 * Purpose     : Packs up, Restore, Index and Pack data files               *
 6 *             : Used for networking application                            *
 7 * Called from: SAMAIN_N.PRG                                                *
 8 * ------------------------------------------------------------------------*
 9 DO WHILE .T.
10 |     @15, 50 CLEAR TO 23,79
11 |     @15, 50, 23, 79 BOX dubl
12 |     @16,57 SAY "Maintenance menu"
13 |     @17, 51 TO 17,78
14 |     * Create a sub menu.
15 |     @18,61 PROMPT "Back up" ;
15 |      MESSAGE "Back up files to floppy diskette(s)"
16 |     @19,61 PROMPT "Restore" ;
16 |      MESSAGE "Restore the data files from floppy diskettte(s)"
17 |     @20,61 PROMPT " Index " ;
17 |      MESSAGE "Rebuild the index files"
18 |     @21,61 PROMPT " Pack  " ;
18 |      MESSAGE "Remove records marked for deletion"
19 |     @22,61 PROMPT "Return " ;
19 |      MESSAGE "Return to the previous menu"
20 |     MENU TO choice4
21 |     *
```

شكل ٣٦-١٢ برنامج SAUTIL_N.PRG

```
22 |     msg = "This option will take between 1 and 20 minutes"
23 |     yn = " "
24 |     DO CASE
25 |     CASE choice4 = 0
26 <--------RETURN
27 |     CASE choice4 = 1
28 |    |    DO saback
29 |     CASE choice4 = 2
30 |    |    DO sarest
31 |     CASE choice4 = 3
32 |    |    DO saindex
33 |     CASE choice4 = 4
34 |    |    DO sapack
35 |     CASE choice4 = 5
36 <--------RETURN
37 |     ENDCASE
38 ENDDO
39
40 *************************
41 *  End of program       *
42 *************************
43
44 *--------------------------------*
45 *   Procedures and Functions     *
46 *   Called only by SAUTIL_N.PRG  *
47 *--------------------------------*
48
49 ****************
50 *  Procedure    : saback
51 *  Purpose      : Backs up data files to floppy diskette(s)
52 ****************
53 PROCEDURE saback
54 SAVE SCREEN
55 CLEAR
56 @ 1,2 SAY msg
57 @ 2,2 SAY  "Do you have formated diskette(s) for back up? [Y/N]" GET yn
58 READ
59 IF UPPER(yn) = "Y"
60 |    source = "C:"
61 |    target = "A:"                  && Target drive for backup
62 |    files1 = source+"CUSTMR.*"
63 |    files2 = source+"INV*.*"       && To include all invoice and
64 |    && inm item files
65 |    RUN BACKUP &files1 &target
66 |    RUN BACKUP &files2 &target/A   && /A Append those files to existing
67 |    && files on the same drive
68 |    ??CHR(7)
69 |    CLEAR
70 |    WAIT ;
70 |     "End backing up ...Remove your diskettes and press any key "
71 ENDIF   (yn = "Y")
72 RESTORE SCREEN
73 RETURN
74
```

تابع شكل ٣٦-١٢

```
 75 *****************
 76 *  Procedure     : sarest
 77 *  Purpose       : Restores data files from floppy diskette(s)
 78 *****************
 79 PROCEDURE sarest
 80 SAVE SCREEN
 81 CLEAR
 82 @ 1,2 SAY msg
 83 @ 2,2 SAY  "Do you have your diskette(s) for restore? [Y/N]" GET yn PICT "!"
 84 READ
 85 IF UPPER(yn) = "Y"
 86 |    source = "A:"                  && Source drive for restore
 87 |    target = "C:"                  && Target drive to accept files
 88 |    files1 = target+"*.*"          && All files in drive A: will be
 89 |    && .DBF, .DBT and .NTX
 90 |    RUN RESTORE &source &files1
 91 ENDIF (yn = "Y")
 92 RESTORE SCREEN
 93 RETURN
 94
 95 *****************
 96 *  Procedure     : saindex
 97 *  Purpose       : Creates or reindexes index files
 98 *****************
 99 PROCEDURE saindex
100 SAVE SCREEN
101 IF use_net("custmr",.T.)    && You need it EXCLUSIVEly
102 |    @24,0
103 |    @24,20 SAY "Please wait...Rebuilding CUSTMR.NTX"
104 |    IF .NOT. FILE ("custmr.ntx")
105 |    |    INDEX ON acct TO custmr
106 |    ELSE
107 |    |    SET INDEX TO custmr
108 |    |    REINDEX
109 |    ENDIF
110 |    @24,20 SAY "Please wait...Rebuilding NAME.NTX"
111 |    IF .NOT. FILE ("name.ntx")
112 |    |    INDEX ON fnam+mnam+lnam TO name
113 |    ELSE
114 |    |    SET INDEX TO name
115 |    |    REINDEX
116 |    ENDIF
117 ELSE
118 <----RETURN
119 ENDIF
120 IF use_net ("invoice",.T.)
121 |    @24,0
122 |    @24,20 SAY "Please wait...Rebuilding INVOICE.NTX"
123 |    IF .NOT. FILE ("invoice.ntx")
124 |    |    INDEX ON inv TO invoice
125 |    ELSE
126 |    |    SET INDEX TO invoice
127 |    |    REINDEX
128 |    ENDIF
```

تابع شكل ٣٦-١٢

```
129 ELSE
130 <----RETURN
131 ENDIF
132 IF use_net ("inv_item",.T.)
133 |     @ 24,0
134 |     @24,20 SAY "Please wait...Rebuilding INV_ITEM.NTX"
135 |     IF .NOT. FILE ("inv_item.ntx")
136 |     |     INDEX ON inv_no TO inv_item
137 |     ELSE
138 |     |     SET INDEX TO inv_item
139 |     |     REINDEX
140 |     ENDIF
141 ELSE
142 <----RETURN
143 ENDIF
144 RESTORE SCREEN
145 RETURN
146
147 ****************
148 *  Procedure     : sapack
149 *  Purpose       : Packs data files
150 ****************
151 PROCEDURE sapack
152 SAVE SCREEN
153 IF use_net("custmr",.T.)   && You need it EXCLUSIVEly
154 |     @ 24,0
155 |     @24,22 SAY "Please wait...Packing CUSTMR.DBF"
156 |     PACK
157 ENDIF
158 IF use_net("invoice",.T.)
159 |     @ 24,0
160 |     @24,22 SAY "Please wait...Packing INVOICE.DBF"
161 |     PACK
162 ENDIF
163 IF use_net("inv_item",.T.)
164 |     @ 24,0
165 |     @24,22 SAY "Please wait...Packing INV_ITEM.DBF"
166 |     PACK
167 ENDIF
168 CLOSE DATABASES
169 RESTORE SCREEN
170 RETURN
171
172 ****************************
173 *  End of file SAUTIL_N.PRG *
174 ****************************
```

تابع شكل ٣٦-١٢

```
 1 * ----------------------------------------------------------------*
 2 * Program    : SAUDFS.PRG                                          *
 3 * Author     : Magdi M. Abu Al-Ata                                 *
 4 * Date       : Nov., 12, 1990                                      *
 5 * Purpose    : Those functions are global to all programs and      *
 6 *            : modules. All of them are used in networking.        *
 7 *            : So I put them in a separate file.                   *
 9 * Called from: All modules                                         *
10 * ----------------------------------------------------------------*
11
12 *****************
13 *  Function      : use_net
14 *  Purpose       : Trys to open a file shared or exclusive for network
15 *  Syntax        : use_net (datfil, mode)
16 *  Parameters    :
17 *      datfil    : File name (must be surrounded in quotes)
18 *      use_mode  : .T. ( = EXCLUSIVE ) or .F. ( = SHARED )
19 *  Example       : IF use_net("custmr", .T.)
20 *                :    SET INDEX TO custmr
21 *                : ELSE
22 *                :    ? " File unavailable"
23 *                :    RETURN
24 *                : ENDIF
25 *****************
26 FUNCTION use_net
27 PARAMETERS datfil, use_mode
28 pause = 5
29 DO WHILE .T.
30 |     IF use_mode
31 |     |    USE &datfil EXCLUSIVE
32 |     ELSE
33 |     |    USE &datfil
34 |     ENDIF
35 |     IF .NOT. NETERR()
36 <--------RETURN (.T.)
37 |     ELSE
38 |     |    IF pause > 0
39 |     |    |    INKEY(1)
40 |     |    |    pause = pause -1
41 ^-------------LOOP
42 |     |    ELSE
43 |     |    |    SAVE SCREEN
44 |     |    |    conf = " "
45 |     |    |    @10,20 CLEAR TO 18,60
46 |     |    |    @10,20 TO 18,60 DOUBLE
47 |     |    |    @12,20 SAY "File" + datfil + "cann't be open at this moment"
48 |     |    |    @14,20 SAY "Do you want to continue attempting to open"
49 |     |    |    @16,20 SAY "it for anouther 5 seconds? [Y/N] -->"
50 |     |    |    SET CURSOR OFF
51 |     |    |    WAIT " " TO conf
52 |     |    |    SET CURSOR ON
53 |     |    |    IF UPPER(conf) = "Y"
54 |     |    |    |    pause = 5
55 |     |    |    |    RESTORE SCREEN
56 ^-----------------LOOP
```

شكل ٣٧ - ١٢ برنامج SAUDFS. PRG.

```
57 |    |    |    ELSE
58 |    |    |    |    RESTORE SCREEN
59 <---------------RETURN .F.
60 |    |    |    ENDIF
61 |    |    ENDIF
62 |    ENDIF
63 ENDDO
64
65 ****************
66 *  Function      : rec_add
67 *  Purpose       : Trys to append a blank record at the end of the
68 *                : file. It locks the record. Returns .T. if the
69 *                : append is successful.
70 *  Syntax        : rec_add ( )
71 *  Example       : IF rec_add( )
72 *                :    REPLACE acct WITH m_acct, fnam WITH m_fnam
73 *                : ELSE
74 *                :    ? " Record unavailable"
75 *                :    RETURN
76 *                : ENDIF
77 ****************
78 FUNCTION rec_add
79 pause = 5
80 DO WHILE .T.
81 |    APPEND BLANK
82 |    IF .NOT. NETERR()
83 <--------RETURN .T.
84 |    ENDIF
85 |    IF pause > 0
86 |    |    INKEY(1)
87 |    |    pause = pause - 1
88 ^---------LOOP
89 |    ELSE
90 |    |    SAVE SCREEN
91 |    |    conf = " "
92 |    |    @10,20 CLEAR TO 18,60
93 |    |    @10,20 TO 18,60 DOUBLE
94 |    |    @12,20 SAY "Cann't create a new record at this moment"
95 |    |    @14,20 SAY "Do you want to continue attempting to open"
96 |    |    @16,20 SAY "it for anouther 5 seconds? [Y/N] -->"
97 |    |    SET CURSOR OFF
98 |    |    WAIT " " TO conf
99 |    |    SET CURSOR ON
100 |    |    IF UPPER(conf) = "Y"
101 |    |    |    pause = 5
102 |    |    |    RESTORE SCREEN
103 ^------------LOOP
104 |    |    ELSE
105 |    |    |    RESTORE SCREEN
106 <------------RETURN .F.
107 |    |    ENDIF
108 |    ENDIF
109 ENDDO
110
```

تابع شكل ٣٧ - ١٢

```
111 *****************
112 *  Function      : lock_rec
113 *  Purpose       : Atempts to lock the current record.
114 *  Syntax        : lock_rec ( )
115 *  Example       : IF lock_rec( )
116 *                :    REPLACE acct WITH m_acct
117 *                : ELSE
118 *                :    ? " Record unavailable"
119 *                :    RETURN
120 *                : ENDIF
121 *****************
122 FUNCTION lock_rec
123 pause = 5
124 DO WHILE .T.
125 |    IF RLOCK()
126 <--------RETURN (.T.)
127 |    ENDIF
128 |    IF pause > 0
129 |    |    INKEY(1)
130 |    |    pause = pause - 1
131 `---------LOOP
132 |    ELSE
133 |    |    SAVE SCREEN
134 |    |    conf = " "
135 |    |    @10,20 CLEAR TO 18,60
136 |    |    @10,20 TO 16,60 DOUBLE
137 |    |    @12,20 SAY "Record cann't locked at this moment"
138 |    |    @14,20 SAY "Do you want to continue attempting to open"
139 |    |    @16,20 SAY "it for anouther 5 seconds? [Y/N] -->"
140 |    |    SET CURSOR OFF
141 |    |    WAIT " " TO conf
142 |    |    SET CURSOR ON
143 |    |    IF UPPER(conf) = "Y"
144 |    |    |    pause = 5
145 |    |    |    RESTORE SCREEN
146 `-------------LOOP
147 |    |    ELSE
148 |    |    |    RESTORE SCREEN
149 <-------------RETURN .F.
150 |    |    ENDIF
151 |    ENDIF
152 ENDDO
153
154 *****************
155 *  Function      : lock_fil
156 *  Purpose       : Atempts to lock the current file.
157 *  Syntax        : lock_fil ( )
158 *  Example       : IF lock_fil( )
159 *                :    DELETE ALL FOR inv_no = m_inv
160 *                : ELSE
161 *                :    ? " File unavailable"
162 *                :    RETURN
163 *                : ENDIF
164 *****************
```

تابع شكل ٣٧ ـ ١٢

```
165 FUNCTION lock_fil
166 pause = 5
167 DO WHILE .T.
168 |     IF FLOCK()
169 <---------RETURN (.T.)
170 |     ENDIF
171 |     IF pause > 0
172 |     |    INKEY(1)
173 |     |    pause = pause - 1
174 ^---------LOOP
175 |     ELSE
176 |     |    SAVE SCREEN
177 |     |    conf = " "
178 |     |    @10,20 CLEAR TO 18,60
179 |     |    @10,20 TO 18,60 DOUBLE
180 |     |    @12,20 SAY "File cann't locked at this moment"
181 |     |    @14,20 SAY "Do you want to continue attempting to open"
182 |     |    @16,20 SAY "it for anouther 5 seconds? [Y/N] -->"
183 |     |    SET CURSOR OFF
184 |     |    WAIT " " TO conf
185 |     |    SET CURSOR ON
186 |     |    IF UPPER(conf) = "Y"
187 |     |    |    pause = 5
188 |     |    |    RESTORE SCREEN
189 ^-------------LOOP
190 |     |    ELSE
191 |     |    |    RESTORE SCREEN
192 <-------------RETURN .F.
193 |     |    ENDIF
194 |     ENDIF
195 ENDDO
196
197 ****************************
198 *  End of file SAUDFS.PRG  *
199 ****************************
```

تابع شكل ٣٧ - ١٢

تعريب النظام

سنشرح فيما يلي كيفية تعريب النظام باستخدام واحد من أشهر برامج تعريب مدخلات ومخرجات الحاسب وهو مساعد العربي/٢ وهو من إنتاج شركة سعودي سوفت.

طورت الشركة برنامج Application Programming Interface وتختصر هكذا (API). بغرض تسهيل التعامل مع مساعد العربي/٢ من خلال البرامج التطبيقية التي يعدها المبرمجون. وهذا البرنامج موجود ضمن ملفات قرص المساندة ويتم نقله إلى القرص الثابت أو المرن أثناء تركيب مساعد العربي/٢ إذا اتبعت تعليمات التركيب التي تنصح بها الشركة المنتجة تحت دليل اسمه: \MA20\API

ويتم التعامل مع برنامج API بإحدى ثلاث طرق:

١ - إما بربط ملف API.OBJ مع البرامج التطبيقية التي يمكن ربطها لاستخراج ملف جاهز للتنفيذ مثل «كلبر» أو «كوبول».

٢ - بتحميل ملف API.BIN في الذاكرة مع البرامج التي لا يمكن ربطها مثل برامج dBASE III PLUS أو dBASE IV

٣ - باستدعاء ملف API.COM مباشرة تحت DOS للحصول على وظائف التعريب للشاشة أو الطابعة... الخ.

ويهمنا من هذه الملفات جميعا ملف CLIP_API.OBJ الذي يستخدم لتعريب تطبيقات «كلبر». ويتم استدعاء هذا الملف بعد ربطه مع النظام المطلوب للتنفيذ بالصيغة التالية:

```
CALL   CLIP_API   WITH   "<functions>"
```

حيث: <function> هي الوظيفة التي تتحكم في برنامج CLIP_API

ملاحظة: سنوضح أهم هذه الوظائف واختياراتها بعد قليل إلا أننا ننصحك بمراجعة كتيب الشركة المنتجة لمساعد العربي/٢ لمزيد من التفصيلات عن الوظائف الأخرى.

فمثلا لكي تستدعي هذا البرنامج ليقلب اتجاه الشاشة من اليمين إلى اليسار مع

عدم تمكين قلبها إلى الاتجاه المعاكس. يتم استدعاء البرنامج هكذا

CALL CLIP_API WITH "SCREEN ARABIC, LOCK"

ونوضح فيما يلي طريقة ربط هذا البرنامج مع النظام الذي بين أيدينا

Plink86 fi SAMAIN, CLIP_API lib CLIPPER, EXTEND

ويشتمل شكل ١٢-٣٨ على برنامج MAIN_ARB.PRG وهو تعديل لبرنامج SAMAIN.PRG ويشتمـل على إجـرائـين جديدين: الأول ARB_MOD ليقـوم بالـوظائف الأساسية التي تتيح تعريب النظام باستخدام ملف CLIP_API في بداية النظام، والثاني LAT_MOD ليعيد اختيارات التعريب التي اختيرت في الاجراء الأول إلى الوضع اللاتيني. والبرنامج يفترض أننا ربطنا ملف CLIP_API.OBJ مع النظام أثناء استخدام Plink86. سنوضح بعد قليل كيفية ربط هذا الملف مع ملفات النظام. ويشتمل شكل ١٢-٣٩ على القائمة الرئيسية للنظام بعد تعريبها والتي تنتج من برنامج MAIN_ARB ولكي تحصل على تعريب كامل للنظام يجب أن تعيد كتابة النصوص التي يشتمل عليها النظام باستخدام محرر السطور الذي يروق لك.

```
 1 * ---------------------------------------------------------------*
 2 * Program     : MAIN_ARB.PRG                                      *
 3 * Author      : Magdi M. Abu Al-Ata                               *
 4 * Date        : October 24, 1990                                  *
 5 * Purpose     : Main Module for arabizing main menu (or the system)*
 6 * Copyright   : Magdi M. Abu Al-Ata                               *
 7 * Parameters  : From DOS, you can optionally pass any parameter   *
 8 *               to run in color mode.                             *
 9 *               missing a parameter will run in  monochrome mode  *
10 * ---------------------------------------------------------------*
11 PUBLIC hd1,hd2
12 * Determine whether the user wants to use color or monochrome monitor
13 IF PCOUNT() > 0
14 |    PUBLIC syscolor
15 |    syscolor = "W+/B+,GR+/R+"
16 |    SETCOLOR(syscolor)
17 ENDIF
18 * Call the arabization module
19 DO arb_mod
20 *  Set clipper environment
```

شكل ٣٨ - ١٢ برنامج MAIN ARB. PRG

```
21 SET SCOREBOARD OFF
22 SET WRAP ON
23 SET ESCAPE OFF
24 SET MESSAGE TO 24 CENTER
25 SET DELETED ON
26 SET CONFIRM ON
27 SET DATE BRITISH
28 SET CENT ON
29 * Save the DOS screen
30 SAVE SCREEN TO dos_scr
31 CLEAR
32 dubl = CHR(201)+CHR(205)+CHR(187)+CHR(186)+CHR(188)+CHR(205) ;
32  +CHR(200)+ CHR(186)+CHR(176)
33 @  2, 14 TO 5, 66 DOUBLE
34 hd1 =  "اسم الشركة :شركة الحاسب العربى"
35 hd2 =  "جميع الحقوق محفوظة ، مجدى ابوالعطـــا"
36 @ 3,t_cent(hd1,80) SAY hd1
37 @ 4,t_cent(hd2,80) SAY hd2
38 @ 0,0
39 @ 0,2  SAY ":التاريخ: "+ DTOC(DATE())
40 @ 0,65 SAY ":الوقت: "+ TIME()
41
42 @  8, 14, 22, 66 BOX dubl
43 @  9,t_cent("80,"القائمة الرئيسية") SAY "القائمة الرئيسية"
44 @ 10, 15 TO 10, 65
45 DO WHILE .T.
46 |    @ 12,35 PROMPT ";  "العمـــــــلاء
46 |      MESSAGE " اضافة - حذف - تعديل فى بيانات العملاء"
47 |    @ 14,35 PROMPT "; "الفواتيـــــــر
47 |      MESSAGE "اضافة أو حذف أوتعديل بيانات فاتورة"
48 |    @ 16,35 PROMPT "; "التقاريـــــــر
48 |      MESSAGE " ارسال التقارير الى الطابعة "
49 |    @ 18,35 PROMPT "; "صيانة الملفات
49 |      MESSAGE "نسخ احتياطية ، استرجاع ، ترتيب ، تنظيف الملفات"
50 |    @ 20,35 PROMPT "; "الخـــــــروج
50 |      MESSAGE "انهاء البرنامج"
51 |    MENU TO choice1
52 |    DO CASE
53 |    CASE choice1 = 1
54 |    |    SAVE SCREEN TO menu_scr
55 |    |    menu_clr = SETCOLOR ()
56 |    |    @24,0 CLEAR
57 |    |    DO sacust
58 |    |    SETCOLOR (menu_clr)
```

تابع شكل ٣٨ - ١٢

```
 59 |   |    RESTORE SCREEN FROM menu_scr
 60 |   CASE choice1 = 2
 61 |   |    SAVE SCREEN TO menu_scr
 62 |   |    menu_clr = SETCOLOR()
 63 |   |    @24,0 CLEAR
 64 |   |    DO sainv
 65 |   |    SETCOLOR (menu_clr)
 66 |   |    RESTORE SCREEN FROM menu_scr
 67 |   CASE choice1 = 3
 68 |   |    SAVE SCREEN TO menu_scr
 69 |   |    menu_clr = SETCOLOR ()
 70 |   |    @24,0 CLEAR
 71 |   |    DO sarep
 72 |   |    SETCOLOR (menu_clr)
 73 |   |    RESTORE SCREEN FROM menu_scr
 74 |   CASE choice1 = 4
 75 |   |    SAVE SCREEN TO menu_scr
 76 |   |    menu_clr = SETCOLOR()
 77 |   |    @24,0 CLEAR
 78 |   |    DO sautil
 79 |   |    SETCOLOR (menu_clr)
 80 |   |    RESTORE SCREEN FROM menu_scr
 81 |   CASE choice1 = 5
 82 |<-------EXIT
 83 |   ENDCASE
 84 ENDDO
 85 RESTORE SCREEN FROM dos_scr
 86 * Call the latin module to reset the arabization options
 87 DO lat_mod
 88 SET COLOR TO
 89 QUIT
 90
 91 *************************
 92 *  End of main program  *
 93 *************************
 94
 95 *------------------------------*
 96 *   Procedures and Functions    *
 97 *   Called only by MAIN_ARB.PRG *
 98 *------------------------------*
 99
100 ****************
101 *  Function      : t_cent()
102 *  Purpose       : Displays text centered on the screen
```

تابع شكل ٣٨-١٢

```
103 *  Syntax          : t_cent (tstring, tlen)
104 *  Parameters      :
105 *      tstring     : Text to be displayed
106 *      tlen        : the width you need text to be centered on
107 *  Example         : t_cent( "Clipper world!", 40 )
108 *****************
109 FUNCTION t_cent
110 PRIVATE tstring, tlen
111 PARAMETERS tstring, tlen
112 STORE INT(tlen/2 - (LEN(tstring)/2)) TO bigcol
113 RETURN bigcol
114
115 *****************
116 *  Procedure       : arb_mod
117 *  Purpose         : To call CLIP_API with arabization options
118 *****************
119 PROCEDURE arb_mod
120 CALL CLIP_API WITH 'CODEPAGE ARABIC'
121 CALL CLIP_API WITH 'NUMER HINDU'
122 CALL CLIP_API WITH 'SCREEN ARABIC'
123 CALL CLIP_API WITH 'ACSD ON'
124 CALL CLIP_API WITH 'DISPLAY VIRTUAL'
125 CALL CLIP_API WITH 'PRINT_ORIEN ARABIC'
126 CALL CLIP_API WITH 'PRIM_LANG NO'
127 * ---- Notes:
128 * You may set MA/2 environment by loading a pre-defined macro
129 * with the following syntax:
130 *  CALL CLIP_API WITH 'MACRO <MA macro filename>'
131 * ----
132
133 *****************
134 *  Procedure       : lat_mod
135 *  Purpose         : To call CLIP_API to reset arabization options
136 *****************
137 PROCEDURE lat_mod
138 CALL CLIP_API WITH 'NUMER ARABIC'
139 CALL CLIP_API WITH 'SCREEN LATIN'
140 CALL CLIP_API WITH 'PRINT_ORIEN LATIN'
141 CALL CLIP_API WITH 'PRIM_LANG YES'
```

تابع شكل ٣٨ - ١٢

التاريخ: ١٨/١٢/٩٠ — الوقت: ٥٣ : ٤٠ : ٠٩

اسم الشركة: شركة الحاسب العربي.
جميع الحقوق محفوظة. مجدي أبو العطا

إضافة ـ حذف ـ تعديل في بيانات العملاء

شكل ٣٩ ـ ١٢ القائمة الرئيسية للنظام العربى

ونـوضـح فيما يلي الـوظـائف التي يشتمـل عليهـا الاجـراءان ARB_MOD و LAT_MOD لتعريب النظام واختياراتها.

الوظيفة	اختياراتها
CODEPAGE	ARABIC: تحدد أن نوع الشفـرة المستخـدمـة هو ٧٨٦ عند استخدام الحروف العربية. ENGLISH: تحدد أن نوع الشفـرة المستخدمة هو ٤٣٧ عند استخدام الحروف الانجليزية.
NUMERALS	HINDU: لاظهار الأرقام بالرسم العربي الموجود على لوحة المفاتيح. ARABIC: لاظهـار الأرقـام بالرسم الانجليزي الموجود على لوحة المفاتيح.

الوظيفة	اختياراتها
SCREEN	ARABIC: تغير اتجاه الشاشة من اليمين إلى اليسار. LATIN: تغير اتجاه الشاشة من اليسار إلى اليمين. لاحظ أن لوحة المفاتيح تتبع اتجاه الشاشة ما لم تغير ذلك.
ASCD	ON: لاظهار الحروف العربية بالرسم العربي الصحيح عند إدخال بيانات. OFF: تلغي اختيار الرسم العربي للحروف العربية الموجودة في لوحة المفاتيح.
DISPLAY	VIRTUAL: لاظهار الحروف العربية عند استخدام الشفرة ٧٨٦. PHYSICAL: لاظهار الحروف الانجليزية عند استخدام الشفرة ٤٣٧.
PRINT_ORION	ARABIC: لتوجيه الطابعة لتطبع من اليمين إلى اليسار. LATIN: لتوجيه الطابعة لتطبع من اليسار إلى اليمين.
PRIM_LANG	NO: لا تسمح بإعادة لوحة المفاتيح للكتابة باللغة الأصلية بعد ضغط مفتاح الادخال أو أحد مفاتيح الوظائف الأخرى. عند التغيير إلى اللغة الأخرى. OFF: تسمح بإعادة لوحة المفاتيح إلى اللغة الأصلية المستخدمة في الكتابة بعد ضغط مفتاح الادخال أو أحد مفاتيح الوظائف عند التغيير إلى اللغة الأخرى.

استخدام ملف تجميعي (Batch file) لترجمة وربط النظام واستخراج ملف جاهز للتنفيذ

يشتمل ملف SMAIN.BAT (شكل ٤٠-١٢) على أوامر ترجمة وربط برامج النظام بحيث ينتج لكل ملف من نوع PRG. ملف مقابل من نوع OBJ. وفي هذا الملف تلاحظ أننا استخدمنا ملف من نوع LNK. للتحكم في اختيارات ربط هذه البرامج جميعا بعد ترجمتها واسم هذا الملف هو SALINK.LNK وهذا الملف يقرأه برنامج الربط Plink86 نتيجة هذا الأمر

Plink86 @salink

ويشتمل شكل ٤١-١٢ على محتويات ملف SALINK.LNK وهي الاختيارات التي تتحكم في برنامج الربط Plink86.

```
echo off
cls
echo *-------------------------------------------------------------*
echo *  This process will compile and link the single user system  *
echo *  It assumes your clipper files are in the sub-directory     *
echo *  \CLIPPER and also assumes it is included in your DOS path. *
echo *  If not, Press Ctrl-C to abort and modify this file and     *
echo *  the SALINK.LNK.                                            *
echo *  Change file names to the correct ones if you want to compile *
echo *  network or arabized system. Add SAUDFS.PRG with network.   *
echo *-------------------------------------------------------------*
pause
clipper samain -m
clipper sainv -m
clipper sainvadd -m
clipper sainvedt -m
clipper sainvdel -m
clipper sacust -m
clipper sacustad -m
clipper sacusted -m
clipper sacustde -m
clipper sarep -m
clipper sautil -m.
plink86 @salink
echo     Type SAMAIN   to start the program
```

شكل ٤٠ - ١٢ ملف S'MAIN.BAT

```
#
# This assumes that your clipper libraries are in subdirectory
# \CLIPPER.  Modify this path if not.
# Change the file names to the correct ones if you want to link
# network or arabized system.
# Add DEBUG.OBJ file for debugging purose.
#
FI SAMAIN
FI SAINV
FI SAINVADD
FI SAINVEDT
FI SAINVDEL
FI SACUST
FI SACUSTAD
FI SACUSTED
FI SACUSTDE
FI SAREP
FI SAUTIL
FI CLIP_API
# Use the following command if CLIP_API.OBJ in \MA20\API directory
# FI \MA20\API\CLIP_API
LIB \CLIPPER\CLIPPER, \CLIPPER\EXTEND
OUTPUT SAMAIN.EXE
```

شكل ٤١ ـ ١٢ ملف SALINK.LNK

تـذكـر...!

شرحنا في هذا الفصل نظاما كاملا لادارة قاعدة البيانات ويشتمل هذا النظام على معظم وظائف نظم إدارة قواعد البيانات وهي الاضافة أو التعديل أو الحذف أو استخراج التقارير أو صيانة الملفات. وقد اعتمدنا في ذلك على فكرة البرامج الصغيرة لكل عملية أو وظيفة لتكون هذه البرامج جزءا من مكتبتك التي تعتمد عليها في بناء وتركيب نظم أخرى لقواعد البيانات ولنوضح لك كيفية تصميم نظم الادارة قواعد البيانـات تعتمد على قائمة رئيسية للنظام وقوائم أخرى تابعة أو منسدلة عنها وإتماما للفائدة شرحنا كيفية تطوير النظام ليستخدم مع شبكة اتصالات محلية والمفاهيم اللازمة لتعريب النظام.

ولأن البرمجة فن أكثر منها علم فإننا نرجو أن نوفق في إعداد نظم لادارة قواعد البيانات لتوافق حاجتك وإمكانياتك.

يخاطب هذا الكتاب كلا من مبرمجي قاعدة البيانات dBASE III PLUS ومن يرغبون في تطوير نظم لادارة قواعد البيانات باستخدام قاعدة البيانات Clipper والكتاب يشتمل على أربعة أبواب على النحو التالي:

الباب الأول: يشرح مفاهيم أساسية عن تاريخ «كلبر» ومتطلباتها وملفاتها وإمكانياتها وضرورة استخدامها في تطوير النظم والفرق بين المفسر والمترجم ويشرح لمبرمجي dBASE III PLUS كيفية توفيق برامجهم قبل ترجمتها باستخدام «كلبر» ويركز على الامكانيات التي يتميز بها «كلبر» عن «دي بيس» في تطوير البرامج والنظم.

الباب الثاني: يشرح مفاهيم متقدمة تهم بصفة أساسية الذين يرغبون في تطوير أنظمة إدارة قواعد البيانات بإمكانيات متقدمة لا توفرها «دي بيس ثري بلاس» مثل المصفوفات واستخدام قوائم الاختيارات ذات الشريط المضاء والتعامل مع شبكات الاتصالات وكيفية التعامل مع أخطاء البرامج وتعقب واكتشاف الأخطاء.

الباب الثالث: يشرح نظاما متكاملا للمبيعات يشتمل على إجراءات وبرامج حية يمكن استخدامها بصورتها الراهنة أو بعد توفيقها لاعداد نظم إدارة قواعد بيانات مشابهة، والنظام يصلح لخدمة مستفيد واحد أو مجموعة مستفيدين داخل شبكة اتصالات محلية.

الباب الرابع: يشتمل على مرجع شامل لجميع الأوامر والوظائف مرتبة ترتيبا أبجديا لسهولة الوصول إلى أي منها، ويشتمل كل أمر أو وظيفة على معلومات وافية تشمل: شرح مختصر، الشكل العام، الاختيارات المتاحة، الشرح، الاختلاف عن «دي بيس ثري بلاس»، مثال على الأقل، الأوامر والوظائف الأخرى ذات الصلة.

- البرمجة باستخدام «كليبر»
- المصفوفات . شبكات الاتصالات . التعامل مع أخطاء البرامج
- مرجع شامل للأوامر والوظائف
- نظام متكامل للمبيعات

طبعت بمطابع الابتكار، هاتف ٨٥٧٢٤٠٤/٨٥٧٢٢٠٤ ـ الدمـام

هذا الكتاب

يخاطب هذا الكتاب كلا من مبرمجي قاعدة البيانات dBASE III PLUS ومن يرغبون في تطوير نظم لادارة قواعد البيانات باستخدام قاعدة البيانات Clipper والكتاب يشتمل على أربعة أبواب على النحو التالي:

الباب الأول: يشرح مفاهيم أساسية عن تاريخ «كلبر» ومتطلباتها وملفاتها وإمكانياتها وضرورة استخدامها في تطوير النظم والفرق بين المفسر والمترجم ويشرح لمبرمجي dBASE III PLUS كيفية توفيق برامجهم قبل ترجمتها باستخدام «كلبر» ويركز على الامكانيات التي يتميز بها «كلبر» عن «دي بيس» في تطوير البرامج والنظم.

الباب الثاني: يشرح مفاهيم متقدمة تهم بصفة أساسية الذين يرغبون في تطوير أنظمة إدارة قواعد البيانات بإمكانيات متقدمة لا توفرها «دي بيس ثري بلاس» مثل المصفوفات واستخدام قوائم الاختيارات ذات الشريط المضاء والتعامل مع شبكات الاتصالات وكيفية التعامل مع أخطاء البرامج وتعقب واكتشاف الأخطاء.

الباب الثالث: يشرح نظاما متكاملا للمبيعات يشتمل على إجراءات وبرامج حية يمكن استخدامها بصورتها الراهنة أو بعد توفيقها لاعداد نظم إدارة قواعد بيانات مشابهة، والنظام يصلح لخدمة مستفيد واحد أو مجموعة مستفيدين داخل شبكة اتصالات محلية.

الباب الرابع: يشتمل على مرجع شامل لجميع الأوامر والوظائف مرتبة ترتيبا أبجديا لسهولة الوصول إلى أي منها، ويشتمل كل أمر أو وظيفة على معلومات وافية تشمل: شرح مختصر، الشكل العام، الاختيارات المتاحة، الشرح، الاختلاف عن «دي بيس ثري بلاس»، مثال على الأقل، الأوامر والوظائف الأخرى ذات الصلة.